Speaker ID with the TIMIT Dataset

Team members: Xinlin Chen, Jessica Centers, and Michael Martinez
External Advisor: Leslie Collins, PhD

December 10, 2019

Abstract

The TIMIT database contains clean audio from 630 speakers reading ten phonetically rich sentences. The goal of this project was to evaluate various neural network (NN) architectures and feature spaces for the purpose of speech-independent speaker identification on this dataset. We evaluate three feature spaces, which we describe as our three methods. The first method was to extract common speech features based on the Mel scale from the audio clips and use those as the input to a neural network. The second method required generating audio spectrograms and using those 2-D representations of the speech as inputs to a neural network. The third method takes an end-to-end deep neural network approach and uses the raw audio data as the input. The use of autoencoders was also explored. For simplicity, the number of speakers considered was reduced to 20, which were randomly selected. Based on our experiments, the best feature space to use in a speaker ID neural network is either common speech features such as Mel filterbanks or the spectrograms of the speech using the Mel scale as these feature spaces provided 85% and 65% accuracy respectively for the 20 speaker subset. Due to the sensitivity of performance with respect to variations in network parameters that members of this project saw and the results reported in some of this project's references, it was concluded that better performance may be possible with either method, but would take an undesirably significant amount of time to tune and train.

1 Introduction

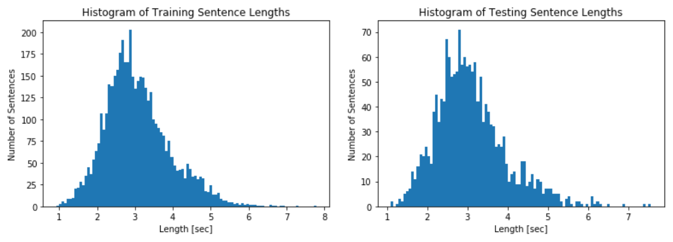
1.1 Motivation and Importance of the Problem

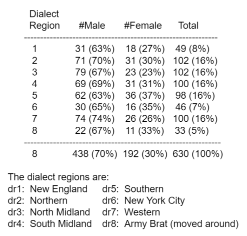
Speaker identification (ID) is critical to authentication and surveillance applications. Example applications include detecting an enemy upon interception of their communication signal, allowing your Amazon Echo or Alexa to identify users that are allowed to command it, or bank account holder verification for transactions being initiated over a phone call. In many scenarios, including over-the-phone listening, speaker identification is limited to only the acoustic-based identification approach. Scenarios in which other biometric measurements would be used to perform speaker identification can easily be argued. For the sake of this project, it is assumed that the acoustic data of a person speaking is the only biometric measurement available and/or favorable for a speaker ID classifier. Initial speaker ID classifiers did not use neural networks (NNs), however, the performance of NN based approaches have surpassed traditional approaches according to the technical literature. For that reason, the goals of this project were to evaluate the performance of different:

1. input feature spaces to a speaker ID neural network.
2. speaker ID neural network architectures for specific input feature.

1.2 Description of the TIMIT Database

The TIMIT Acoustic-Phonetic Continuous Speech Corpus was designed specifically to provide speech data for acoustic-phonetic studies and for the development/evaluation of automatic speech recognition systems. The speech data is sampled at a rate of 16kHz and the speech sentence lengths of both the training and testing subsets can be seen in the histograms below. Also shown is the distribution of speakers by dialect and gender. The TIMIT database is provided





Dialect Region	#Male	#Female	Total
1	31 (63%)	18 (27%)	49 (8%)
2	71 (70%)	31 (30%)	102 (16%)
3	79 (67%)	23 (23%)	102 (16%)
4	69 (69%)	31 (31%)	100 (16%)
5	62 (63%)	36 (37%)	98 (16%)
6	30 (65%)	16 (35%)	46 (7%)
7	74 (74%)	26 (26%)	100 (16%)
8	22 (67%)	11 (33%)	33 (5%)
8	438 (70%)	192 (30%)	630 (100%)

The dialect regions are:

dr1:	New England	dr5:	Southern
dr2:	Northern	dr6:	New York City
dr3:	North Midland	dr7:	Western
dr4:	South Midland	dr8:	Army Brat (moved around)

Figure 1: Statistical graphics of TIMIT database characteristics.

in training and testing subsets, however the speech was combined and re-divided into these subsets for this project because the original divide has no speaker in both the training and testing subsets. Since each speaker contributes 10 sentences to the database, sentences per speaker were separated into training and testing sentences such that there were 7 sentences in training and 3 in testing

[10]. For use of the TIMIT database, the TIMIT utility library was used. The code provided through this library allowed us to easily load, parse, and use the TIMIT database [12].

1.3 Overview of Methods

Speech features can provide insight as to who the speaker is, what the speaker is trying to say, and what state the speaker is in. To identify who the speaker is, features stemming from the differences in human vocal tracts, the wording choices people make, or the accent the the speakers talks in are vital. Similarly, extracting specific phonemes and overall words are more important to identifying the content of speech. Then lastly, identifying what state the speaker is in is typically done by extracting prosodic features describing the tone, rhythm, and emphasis of the speech.

Our group explored three feature spaces we believed could best preserve speaker-dependent features. These feature spaces are described below:

1. Mel-scale based features described in the technical literature were extracted from each speech data segment and used as the input to a neural network. Common Mel-scale based features are the Mel-frequency cepstral coefficients, log Mel filterbank coefficients, and delta and delta-delta coefficients.

2. Speech was processed into a 2-D representation that is its spectrogram and that 2-D representation for each the speech data segments was used as the input to the neural network. Various frequency-time representations were considered in creating the spectrograms.

3. The time-series data or simple transforms of the time-series data for each speech data segment was used as the input to several different neural networks. The time-series data was also fed into several autoencoders to reduce dimensionality before using a classification network.

2 Related Works

The literature used for each method, as described in the section above, are described below.

2.1 Method 1: Related Works

This method was inspired by the paper presenting the Deep Speaker, a neural speaker embedding system that was designed to perform speaker identification and verification [7].

The Mel scale was originally derived to measure the subjective pitch of tone [14]. The scale is proportional to the perceived magnitude of subjective pitch [15], taking into account the human ear's ability to discriminate frequency intervals at lower and higher frequencies [14]. Mel scale-based features leverage this scale, and are popular features in automatic speech and speaker recognition algorithms; some examples of these features are log Mel filterbank coefficients, Mel frequency cepstral coefficients, delta and delta-delta coefficients [17, 6]. While all of these features were used in the baseline model in the Deep Speaker paper, the inputs to the final proposed model only consisted of log Mel filterbank coefficients [7].

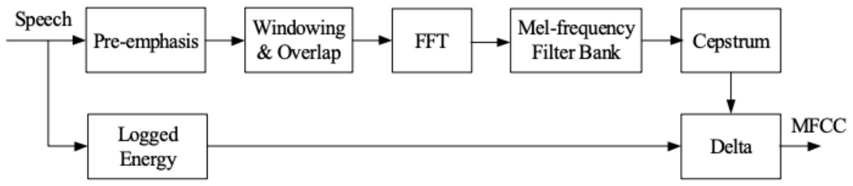


Figure 2: Block diagram of MFCC extraction procedure, taken from [18].

The Deep Speaker paper explored two neural networks: a ResNet-based deep CNN ('ResCNN') and an architecture with convolutional layers followed by gated recurrent unit layers [7]. The first approach was explored for this method.

In the paper, datasets with as many as 250 utterances per speaker were employed. However, the TIMIT database is more limited in this scope, containing the audio of only 10 sentences per speaker. Therefore, the proposed architecture from this paper was scaled down for this method. ResNet employs a sequence of stacked residual blocks, or 'ResBlocks'. In this paper, the ResBlock consists of 2 convolutional layers, each with a kernel size of 3, a stride of 1, and contains a skip connection from the input of the ResBlock. The ResCNN contained four repeated units with three stacked 'ResBlocks' each, and a total number of about 23-24 million parameters. Due to worries of overfitting, only one 'ResBlock' is included in this method. The combined number of parameters in the initial convolution layer and the ResBlock, with 64 hidden channels, is approximately 252,000 parameters. Additionally, in the paper, ReLu was used as the nonlinearity for all network layers. In this method, leaky ReLu is employed for reasons discussed in section 3.2.

Multi-task learning was also employed to improve results, due to the small size of the dataset. Separate fully connected layers connected to separate cross-entropy loss functions were used to let the model learn to derive the speakers' identity, gender, and region. However, while the TIMIT dataset contains data from speakers originating from 8 regions, the model will be trained and tested on only 20 speakers. For this reason, while speakers were randomly picked from the dataset, the speakers' regions were restricted to a random 3 regions out of the original 8.

2.2 Method 2: Related Works

Since the goal of this project is to identify speakers, the the most desirable feature space for the input to our neural network is one that one falls within the first category. One of the difficulties with speech, however, is that speech features are not necessarily separable by those three categories. To best avoid training the network to be speech dependent, speech samples were chosen randomly within the full length of speech provided for each speaker and a convolutional neural network (CNN) was used to capture features seen across various words from a speaker. Prosody from emotion were avoided in this project because the TIMIT dataset was collected by having speakers read sentences which were not designed to be emotionally evoking. It should be noted that prosody from accent is still seen within the dataset.

Additionally, since some of the most speaker-specific features are seen in the frequency domain, the inputs to the CNN were speech spectrograms. Using the spectrograms of speech data as the input to a speaker identification system have been considered by many for over 50 years not [2]. In this method, spectrograms are the broad term used for a 2-D, frequency-time representation of speech. Three different time-frequency representations were considered.

1. The most basic spectrogram is that generated from a Short-Time Fourier Transform (STFT). This transform must defined by a window length, window overlap, and window type. The speech processing community often uses a window length between 20-30 ms. When the STFT is used in this project, a rectangular window of 20 ms is used with 50% overlap was used.

2. The next spectrogram used in this project was the spectrogram associated with the Librosa Python library. This spectrogram differs from the previous only in that the frequency axis is now in the Mel scale. This is the spectrogram used in the paper for which the architecture in this method was based on [8].

3. The last spectrogram actually a continuous wavelet transform (CWT). There are a number of wavelets that were considered in this method, but the one that was primarily experimented with was the Gaussian Derivative Wavelet of order P, where P was changed from 2 to 8. Although most references indicate that CWT can be a better transform for speech detection than traditional spectrograms, it was still worth experimenting with for this project as references such as [3] describe at a transform to easily find speech pitch.

2.3 Method 3: Related Works

There are relatively few speaker recognition papers that use raw audio data as the input. Most of the speaker recognition papers extract features prior to using a NN. Of the papers that use raw audio data as an input, most of these methods are text-dependent systems used for authentication [5]. This is an important distinction since there is generally a known passphrase that is spoken and training and testing is performed on this phrase only. Text-independent systems are harder to train using raw data since many different phrases of varying length must be accounted for. In this case, one would hope that a NN would learn speech features that are at least partially invariant to the phrases being spoken. For text-independent speaker recognition, the training data and how it is pre-processed and split up becomes very important.

Recently, raw audio data has been modeled with deep NNs in a way that is useful for speaker recognition [9, 11, 13, 16]. Muckenhirn et al. use an approach where a convolution neural network (CNN) and multilayer perceptron (MLP) are first trained to classify unknown speakers, then the network is adapted to build a speaker-specific binary classifier for speaker verification. The high level overview of this system is shown in Figure 3. By analyzing the filters in the

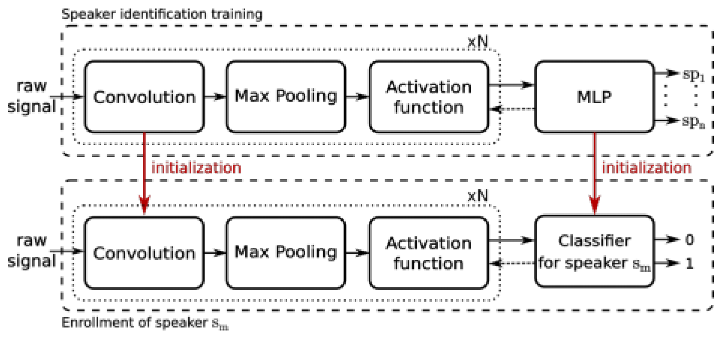


Figure 3: High level overview of the approach for speaker verification used in [9].

first convolutional layer, Muckenhirn et al. found that the CNN emphasizes frequencies below 1000 Hz and models fundamental frequency information. Since these filters can be used to emphasize "unique" frequency components related to how an individual speaks, they are somewhat phrase invariant features. However, not all spoken phrases have the same frequency content so it still matters what phrases are being spoken if there is limited testing data. When more testing data is available, these concerns are alleviated. Most of these papers also use voice activity detection to identify when someone is speaking or not and only take training data when speech is detected. This could also help improve our training and testing data, but was not performed for this project.

3 Details of the Project

For all methods that were experimentally evaluated for this project, a few data organization techniques were kept common. Additionally, more traditional classifiers were used as baselines to performance expectations.

3.1 Data Splitting

Throughout the literature review, we found the number of seconds considered per data segment varied. For this reason, the different methods of this project considered a number of segment lengths experimentally. For generalization, say data segments were desired to be L seconds long.

An additional parameter was introduced to indicate how many L-second speech segments should be randomly selected from a speaker. This parameter n was calculated based on the average number of seconds a speaker collectively had within the training and testing subsets. This average length of speech data per speaker was 31 seconds. Then, n was selected to be some integer multiple of 31. The results of each method will describe what integer multiple was used. In general, selecting n is a matter of trying to have enough samples per speaker in order to do classification while also not over-sampling and therefore causing the neural network to over-fit to the specific sentences within the training subset.

3.2 Scaling for the Number of Speakers

Because speaker identification is not expected to work well for the full 630 speakers in the dataset if it can't work well for fewer speakers, we decided to reduce the number of speakers for experimental purposes. This allowed us to more quickly tune network parameters and explore more network architectures.

It should be noted that in the literature review as well as our experimental results, we saw that by scaling down the number of speakers being used, that the network must also scale down to work properly. This trend is due to the fact that since less speakers equates to less data, the number of weights associated with the deep networks becomes too large to be learned. Quickly, we discovered that if we do not scale down our network, that issues such as vanishing gradients arise and our network would begin predicting only a single class resulting in performance around chance. We were able to use leaky rectified linear unit (ReLu) activation functions in the place of or non-leaky ReLu activation functions and see this behavior improve, but performance was still only 50% better than chance, which for 20 speakers was about 7.5% accuracy. The better solution, we discovered, was to scale our NN down as our number of speakers decreased, as eluded to earlier. This was done through reducing the number of nodes within layers as well as reducing the number of layers within our network.

3.3 Baseline Classifier Performance Results

The baseline classifier results were generated from three naïve approaches. In the first approach, the raw one second audio segments were input to a logistic regression classifier. These audio segments were scaled so the maximum amplitude was 1 so that the effect of people speaking at different volumes was reduced. In the second approach, principal component analysis (PCA) was used to reduce the dimensionality of the time-series data segments from 16000 samples to a feature vector of length 200, then this feature vector was fed into the logistic regression classifier. A feature vector of length 200 was chosen because for 20 speakers this accounted for 75% of the explained variance in the data and the rest of the eigenvalues were small comparatively. In the third approach, the normalized magnitude coefficients of the Discrete Fourier Transform (DFT) of each audio segment were calculated and then the coefficients corresponding to the frequencies from 0 Hz to 2000 Hz were used as the input to the logistic regression classifier. The coefficients for each segment were normalized by scaling the largest coefficient to be equal to 1. The justification for using these coefficients as features is discussed in Section 5.3.

The first and second approaches both failed in the same way. The logistic regression model failed to generalize from the training data to the testing data. Despite attempts to optimize the learning rate, momentum, and regularization, the testing accuracy would never go above 5-8% for 20 speakers. The results presented here are only for the case using the time-series data since the results using PCA first were almost exactly the same. As shown in Figure 4, the cross entropy loss for training and testing would stay relatively constant. The training accuracy would often increase steadily and plateau in the 90% range or exhibit behavior similar to that shown in Figure 4.

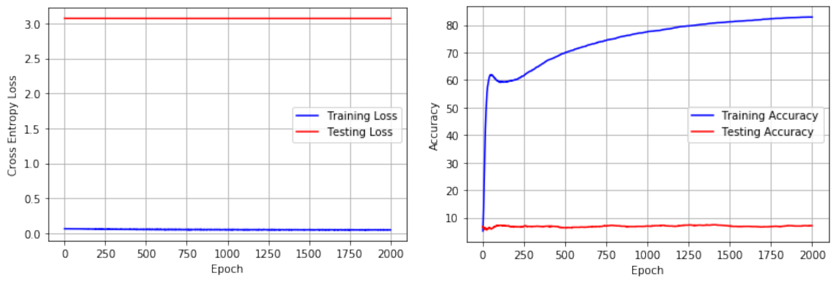


Figure 4: The cross entropy loss (left) and accuracy (right) for the logistic regression classifier using the audio time-series segments as inputs.

The third method, using 2049 normalized DFT magnitude coefficients as the input to the logistic regression classifier, performed the best. The testing accuracy converged to approximately 50% while the training accuracy converged to approximately 85%. The accuracy and loss curves can be seen in Figure 5. Since these were all basic, naïve approaches, the goal for the methods outlined

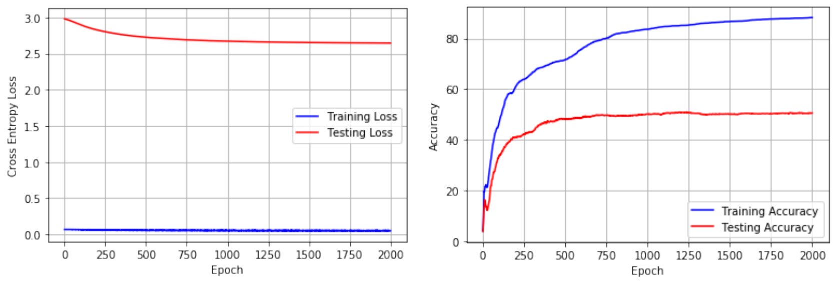


Figure 5: The cross entropy loss (left) and accuracy (right) for the logistic regression classifier using the audio time-series segments as inputs.

below is to beat this performance. Keep in mind that there is limited data for each speaker so it may be difficult to train a neural network.

3.4 Multi-Task Learning

It has been demonstrated, both theoretically and empirically, that learning multiple tasks at the same time tends to greatly improve classification performance on the tasks over learning each task independently [1] when there are relations between the tasks [4]. This is especially relevant when limited observations of the data are available, which is the case here, as we only have ten utterances per speaker. It becomes advantageous to 'pool' information across these tasks [1]. In the TIMIT database, meta-data is available for each speaker. For our purposes, learning the speaker's gender and region can serve as related tasks when learning the speaker's identity, as they may influence voice pitch and accent.

In methods 1 and 2, three separate fully connected layers are appended to the final layer of the neural network architecture, each separately trained using a loss function comparing the output of the layer to an attribute of the speaker. As discussed above, the speaker's identity, gender, and region are compared. The neural network's overall loss function becomes a weighted linear combination of these functions.

$loss_{total} = A * loss_{id} + B * loss_{gender} + C * loss_{region}$, where $A + B + C = 1$

4 Contribution of Each Member of the Team

This section includes each team member's descriptions of their contribution to this project. All three members contributed to the report document and slides.

4.1 Xinlin Chen

Janet implemented the features and network architecture described in the Method 1 section and compared various training parameters, and introduced the multi-task learning component of Methods 1 and 2.

4.2 Jessica Centers

Jessica implemented everything discussed in the Method 2 sections. Additionally, Jessica set up the TIMIT database and dataloaders for the group to use and implemented the data splitting and sampling technique described in the above section.

4.3 Michael Martinez

Michael contributed by generating the baseline classification performance results as well as implementing and comparing all the different networks used in the Method 3 section.

5 Experimental Results

5.1 Method 1: Results

Features: Log mel-filterbank features were used as an input to the neural network architecture described below. A window length of 0.025 ms was used, and 64-dimensional coefficients were calculated and normalized to have zero mean and unit variance, as stated in the Deep Speaker paper [7].

Overall architecture: the Deep Speaker architecture consists of four Res 'units', each containing a 2D convolutional layer followed by three stacked ResBlocks with the architecture previously described in section 2.1 [7]. The Res units have the following output dimensions: 64, 128, 256, 512. Each Res unit contained roughly quadruple the number of parameters of the previous unit, with the first unit consisting of 252,000 parameters. However, since the paper leveraged a much larger dataset for speaker identification than TIMIT, the depth of the architecture unnecessarily increased training time and was unsuitable for this problem. Therefore, the network was scaled down for this method, and only the first Res 'unit' was used. The first convolutional layer has a kernel size of 5, a stride of 2, and 64 output channels. The output from this layer is ReLU-ed and maximum-pooled with a kernel size of 2, and then inputted to the ResBlock. Each convolutional layer in the ResBlock has a kernel size of 3, a stride of 1, and 64 output channels. To simultaneously train the network on multiple related tasks, fully connected layers mapping the flattened output of the Res unit to speaker metadata (identity, gender, and region) were also incorporated and trained using separate loss functions. Again, to account for the small number of speakers chosen (20), the speakers were only drawn from 3 randomly chosen regions.

Multi-task learning: The inclusion of multi-task learning to the network improved performance by 3%. For this experiment, the model was trained with a batch size of 50 using the Adam optimizer. The loss function used was $loss_{overall} = 0.8 * loss_{identity} + 0.1 * loss_{gender} + 0.1 * loss_{region}$. This loss function was used in lieu of the loss function suggested by the Deep Speaker paper, owing to the difference in number of speakers and observations utilized.

Optimizer: The SGD, Adagrad and Adam optimizers in the torch.optim package were explored, with the Adam optimizer producing the best performance. The optimizer parameters β_1, β_2 and learning rate were adjusted, with the best performance occurring when these parameters were set to 0.9, 0.999, and 1e-3, respectively.

Non-linearity: another modification made from the Deep Speaker architecture was the choice of nonlinear activation function. Due to the vanishing gradient problem, the nonlinear function used after each convolutional layer and the skip connection in the ResBlock was leaky ReLU with a negative slope of 0.1 at x<0. ReLU was used after the initial convolutional layer in the network.

Batch normalization was utilized after each convolutional layer in the Res-Block, as suggested in the Deep Speaker paper.

Various batch sizes were tested with the fine-tuned Adam optimizer parameters. Classification results on the testing set with these batch sizes at 10 epochs, averaged over 5 runs, are shown in Figure 6 on the next page.

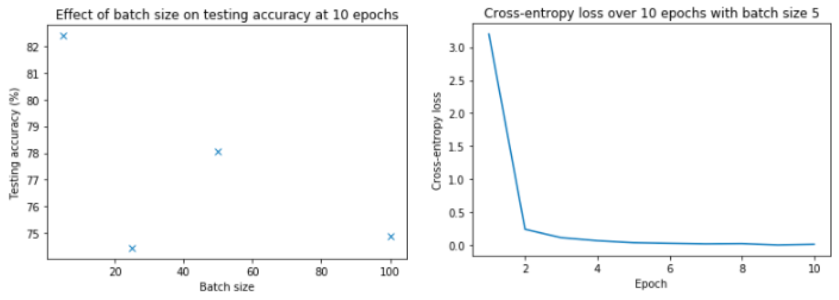


Figure 6: Left: testing accuracy using mel filterbanks and modified ResNet architecture after 10 epochs using different batch sizes and the Adam optimizer. Over 5 runs, a batch size of 5 produced the best results. Right: cross-entropy loss during training for the run with a batch size of 5 that produced the highest testing accuracy (84.03%).

5.2 Method 2: Results

The network architecture considered was modified from [8]. It consists first of a 2-D convolutional layer with kernel size 3, stride 1, and 30 output channels. Those outputs are maximum-pooled with kernel size 2. The next layer is another 2-D convolutional layer with kernel size 3, stride 1, and 15 output channels. That output also is maximum-pooled with kernel size 2. The data is then flattened and fed into a linear layer with 200 nodes. This is followed with a dropout layer of 50% and then three fully connected layers to be used for three different loss calculations associated with speaker ID, speaker gender, and speaker region. The multiple loss functions allow the network to learn quicker by utilizing the additional knowledge of the speaker. A rectified linear unit (ReLu) activation function was used at every layer except for the last layers which have a softmax activation function.

This architecture was constantly manipulated and tested. Things considered include:

- Adding or Removing Layers: The general structure was kept but more convolutional layers at the beginning and linear layers at the end were considered.
- Kernel Sizes and Step Size: Different kernel sizes and step sizes for the convolutional layers were considered. Different kernel sizes for the maximum pooling layers were also considered.
- Batch Size: The batch size appeared to make more of a difference than our group imagined, so various batch sizes were considered and method 2 concluded in using a batch size of 30.

- Learning Rate: This was important to tune so the network converged quick enough, but did not encounter vanishing gradients. The learning rate used in the this method's best performance was 0.005.

- Criteria: Both Adam and stochastic gradient descent (SGD) were considered. SGD with Nesterov momentum seemed to work best.

- Sample per Speaker: Experimentally, it appeared that the larger number of random samples taken per speaker worked best even though the total number of data was finitely 10 sentences worth. Essentially taking more samples, allowed the network to be less fitted to the start point of speech. Method 2 concluded in using 310 samples per speaker. Since the average number of seconds per speaker is 31 seconds, the n described in 'Details of the Project' was 10 for this method.

- Length of Samples: Since the STFT and Librosa spectrograms utilize a moving window of 20 ms, samples were required to be greater than 20 ms. It also seemed appropriate to keep samples at less than half the average sentence length. It was then experimentally determined that 1 second speech samples worked pretty well.

- Multi-Task Learning Allocations: This method is described in 'Details of the Project'. The allocation used in this method's best performance was 90% speaker identity, 5% speaker gender, and 5% speaker region. Therefore the total loss was $loss_{total} = 0.9loss_{id} + 0.05loss_{gender} + 0.05loss_{region}$.

- Spectrogram Type: This was the main consideration in this method. Details can be seen below.

Spectrograms generated via STFT, the Librosa library that utilizes the Mel scale, and CWTs were first compared. For the CWT's spectrograms, the primary wavelet used was the Gaussian Derivative Wavelet with order 2, 4, 6, and 8 which corresponds to the order derivative. The Morlet and Mexican Hat Wavelets were also considered, but the best performance for any CWT was seen for the Gaussian Derivative Wavelet of 4th order. After 10 epochs, it was seen that the Librosa library's spectrograms were the better inputs to the speaker ID neural network.

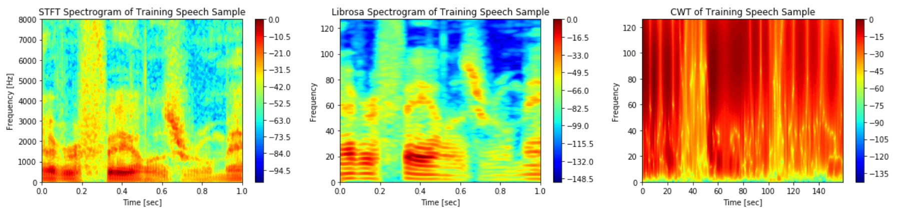


Figure 7: Various spectrograms considered (STFT, Mel-scale STFT, CWT).

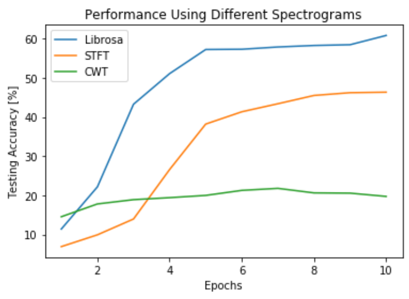


Figure 8: Performance over 10 epochs for using the various spectrogram inputs.

5.3 Method 3: Results

In this section, two different features were used as inputs to the various NNs. These inputs were segments of the raw audio data or the normalized DFT magnitude coefficients corresponding to the frequencies from 0 Hz to 2000 Hz. 2000 Hz was determined as the cutoff by looking at a random selection of the magnitude spectra and noticing that most of the frequency content occurred below 2000 Hz. The point of this section was to determine what kind of classification performance was possible doing as few transformations as possible to the data before feeding it into the network. Taking the DFT and using the magnitude coefficients as features is one of the most simple transformations possible and does not fundamentally alter the information fed into the network. The DFT coefficients were normalized so that the maximum magnitude in each spectrum was 1 to remove effects of speaker volume so that the coefficients more directly comparable between subjects. The approaches and NN architectures discussed below focus on the more successful attempts and omit details such as parameters for the failed approaches.

The approaches for the time-series inputs attempted to use standard feed-forward NNs, CNNs, variational autoencoders (VAEs), and contractive autoencoders (CAEs). Using the audio time-series segments as inputs to NNs for classification generated poor results. These results showed a testing accuracy of approximately 7.5%. The next approach used autoencoders to reduce the dimensionality of the data before feeding the embedded data into a NN for classification. Both a VAE and CAE were used. Two CAE architectures were tried. One reduced the dimensionality from 16000 samples to 400 features and the other reduced the dimensionality to 2000 features. The latter architecture used four layers whose output dimensions were 8000, 6000, 4000, and 2000. The loss, original signal segment, and reconstructed signal segment are shown in Figure 9. As you can see, the reconstructed signal did not resemble the original so classification post dimensionality reduction did not work. Overall, the performance of the time-series based approach was poor.

The results of the DFT coefficient approach were on par with the baseline classifier results. In both cases the batch size was 100, an Adam optimizer

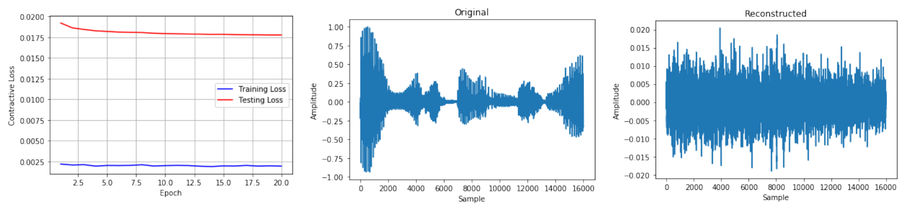


Figure 9: The contractive loss (left), original audio segment (middle), and reconstructed audio segment (right) using the audio time-series as input to the CAE.

was used with a learning rate of $lr = 0.00001$, $\beta = [0.98, 0.999]$, and a weight decay of $wd = 0.00001$, and the loss function was cross entropy loss. These parameters were tuned through trial and error and happened to work well for both the standard feed-forward and CNN architectures. Recall that 2049 DFT coefficients were used. The DFT length was 16384, which is the next power of two greater than the data length of 16000 samples. The first layer of the standard feed-forward architecture had an input size of 2049 and output size of 1600. The next three layers had input and output sizes of (1600, 400), (400, 100), and (100, 20) respectively. The first three layers used a ReLu activation function and the last layer used softmax. The results of the standard feed-forward architecture are shown in Figure 10.

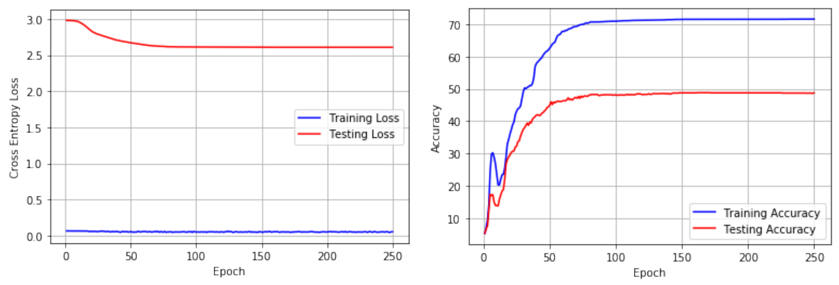


Figure 10: The cross entropy loss (left) and accuracy (right) for the feed-forward NN using the DFT coefficients as inputs.

The architecture of the CNN was as follows. There are four convolutional layers using 32, 24, 16, and 12 filters. For each layer the kernel sizes were 32, 16, 8, and 4. Each convolutional layer used a stride of 1 and no padding. After each convolutional layer max pooling was used with a kernel size of 2. Then four fully connected layers with input and output dimensions of (1452, 1000), (1000, 500), (500, 100), and (100, 20) were used. Each layer except for the last two used a tanh activation function. The last two layers used a ReLu and softmax activation function respectively. The results of this network are shown in Figure 11.

In all of the above architectures, dropout and batch normalization were tried,

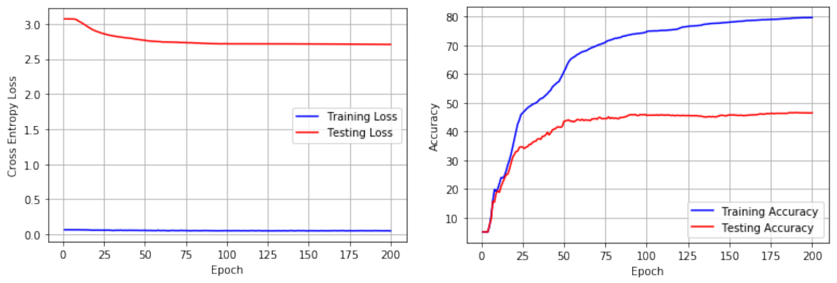


Figure 11: The cross entropy loss (left) and accuracy (right) for the CNN using the DFT coefficients as inputs.

but they did not improve results so they weren't used in the final architectures.

6 Concluding Remarks

Key findings in this project included that it simply takes a lot of data samples to train a deep neural network, how you sample speech is vital to a speaker identification system, parameter tuning takes time and is quite sensitive, multi-task learning can help an underfed network adapt quicker, and that speech data transformed into some sort of frequency representation (ie. DFT coefficients, spectrograms, MFCC coefficients, etc.) tend to be better inputs into a speaker identification system than features extracted in the time domain. Pre-processing the data using voice activity detection so that noise segments between words are removed could also improve training time and overall accuracy.

The results of all three methods described in this project suggest that the best feature space to use in a speaker ID neural network is either common speech features such as mel filterbanks or the spectrograms of the speech using the Mel scale as these feature spaces provided 85% and 65% accuracy respectively for the 20 speaker subset.

Due to the sensitivity of performance with respect to variations in network parameters that members of this project saw and the results reported in some of this project's references, it was concluded that better performance may be possible with either method, but would take an undesirably significant amount of time to tune and train.

References

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 41–48. MIT Press, 2007.

[2] Richard H. Bolt, Franklin S. Cooper, Edward E. David Jr., Peter B. Denes, James M. Pickett, and Kenneth N. Stevens. Identification of a speaker by speech spectrograms. *The American Association for the Advancement of Science*, 166:338–343, 1969.

[3] S. Alenka Brown-VanHoozer. Speaker recognition through nlp and cwt modeling. *15th Annual NDIO Security Technology Symposium Exhibition*, 1999.

[4] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 109–117, New York, NY, USA, 2004. ACM.

[5] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer. End-to-end text-dependent speaker verification. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5115–5119. IEEE, 2016.

[6] Hemant Kumar Kathania, S. Shahnawazuddin, Waquar Ahmad, and Nagaraj Adiga. Role of linear, mel and inverse-mel filterbanks in automatic recognition of speech from high-pitched speakers. *Circuits, Systems, and Signal Processing*, 38(10):4667–4682, Oct 2019.

[7] Chao Li, Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, and Zhenyao Zhu. Deep speaker: an end-to-end neural speaker embedding system, 2017.

[8] Yanick Lukic, Carlo Vogt, Oliver Durr, and Thil Stadelmann. Speaker identification and clustering using convolutional neural networks. *2016 IEEE International Workshop on MAchine Learning for Signal Processing*, 2016.

[9] Hannah Muckenhirn, Mathew Magimai Doss, and Sébastien Marcell. Towards directly modeling raw speech signal for speaker verification using cnns. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4884–4888. IEEE, 2018.

[10] Massachusetts Institute of Technology, SRI International, and Inc. Texas Instrument. Timit acoustic-phonetic continuous speech corpus, 1993. data retrieved from Linguistic Data Consortium, `https://catalog.ldc.upenn.edu/LDC93S1`.

[11] Dimitri Palaz, Ronan Collobert, and Mathew Magimai Doss. Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks. *arXiv preprint arXiv:1304.1018*, 2013.

[12] Colin Prepscius. timit utility library, 2018. data retrieved from Colin Prepscius' Github, `https://github.com/colinator/timit_utils/blob/master`.

[13] Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals. Learning the speech front-end with raw waveform cldnns. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[14] Robert J. Siegel. A replication of the mel scale of pitch. *The American Journal of Psychology*, 78(4):615–620, 1965.

[15] S.S. Stevens, J. Volkmann, and E.B. Newmann. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(185), 1937.

[16] Zoltán Tüske, Pavel Golik, Ralf Schlüter, and Hermann Ney. Acoustic modeling with deep neural networks using raw time signal for lvcsr. In *Fifteenth annual conference of the international speech communication association*, 2014.

[17] R. Vergin, D. O'Shaughnessy, and A. Farhat. Generalized mel frequency cepstral coefficients for large-vocabulary speaker-independent continuous-speech recognition. *IEEE Transactions on Speech and Audio Processing*, 7(5):525–532, Sep. 1999.

[18] Wei Han, Cheong-Fat Chan, Chiu-Sing Choy, and Kong-Pang Pun. An efficient mfcc extraction method in speech recognition. In *2006 IEEE International Symposium on Circuits and Systems*, pages 4 pp.–, May 2006.