Speaker Recognition Using Deep Belief Networks

[CS 229] Fall 2012:12-14-12

Alex Fandrianto
afandria@stanford.edu

Ashley Jin
ashpjin@stanford.edu

Aman Neelappa
aman313@stanford.edu

1 Introduction

Speaker recognition and verification is essential in confirming identity in numerous real-world applications. Speaker recognition is identifying an individual speaker from a set of potential speakers while speaker verification is confirming a speaker's identity as the true speaker or as an imposter who may be trying to infiltrate the system. In speaker recognition and verification, one of the major challenges is choosing good features as inputs to a classifier. Recently, interest in using deep learning methods to learn features from audio data in an unsupervised fashion has grown. The high level representations learned in the higher layers are found to have comparable and often better performance than traditional features such as Mel-Frequency Cepstral Coefficients (MFCC) [1].

In this study, we will explore the benefit of using different features to transform input to a neural network as well as the application of convolutional neural networks in speaker recognition and verification. We will study the results on text independent corpora.

2 Data

For this project, we used two different datasets: TIMIT and the West Point Company G3 American English corpus.

2.1 TIMIT Data

We performed initial experiments using the TIMIT Dataset. TIMIT is comprised of recordings of 630 speakers (438 male, 192 female[2]) using eight dialects of English. Each speaker has ten different utterances over a clean channel (no noise). Each utterance is about 3 seconds and is phonetically diverse. The dataset contains about 5.25 hours of audio in .wav format.

2.2 West Point Company G3 American English Speech Data

We also used the West Point Company G3 corpus. G3 contains recordings from 109 speakers (56 male, 53 female). There are 185 possible utterances that cover all or most possible American English syllables and each speaker recorded 104 of those utterances. Each utterance is about 4 seconds. The dataset contains about 15 hours of audio in .wav format.

We used only the male audio for our experiments.

2.3 Feature Extraction

We used Matlab to extract features from the raw data to input into our neural network. After reading in the wav files, we used matlab to generate a spectrogram. The parameters we gave the spectrogram function were:

-window: 20ms (copied from [1])
-overlap: 10ms (copied from [1])
-frequencies: 512
-sampling rate: 16000/22050
(depending on dataset)

The spectrogram is formatted with time on the y axis and frequency on the x axis. The matrix of spectrogram values is of size number of time windows (with overlap) x number of frequencies. Since the dimensionality is very large and our training data set is small, we extracted features from the spectrogram by reducing the information to a single vector in different ways. We discuss these reductions in the Experiments section.

3 Convolutional Neural Networks

The motivation for convolutional neural networks (CNNs) comes from the working of the cat's visual cortex. Essentially the major ideas can be summed up as:

a) Exploit locality: Instead of using a fully connected network, connect a node to only a few local nodes from the previous layer. Higher-level nodes will use more global information.

b) Feature Maps: Sets of nodes at certain layers are forced to have the same weights connecting to the layer above them. The learned feature maps allow the CNN to transform the signal in a piecewise manner.

A CNN's hidden layers are typically interspersed with "max-pooling" layers. These layers select the max of small non-overlapping windows of the previous activations to send to the next layer. This reduces the computational size of the problem and makes the architecture robust to small translations in the input signal.

Table 1: Train time comparison with and without noise and normalization. Normalization decreases train time in both situations.

	No Noise	Noise
Baseline	840.43 seconds	402.6 seconds
w/Norm	637.54 seconds	351.82 seconds

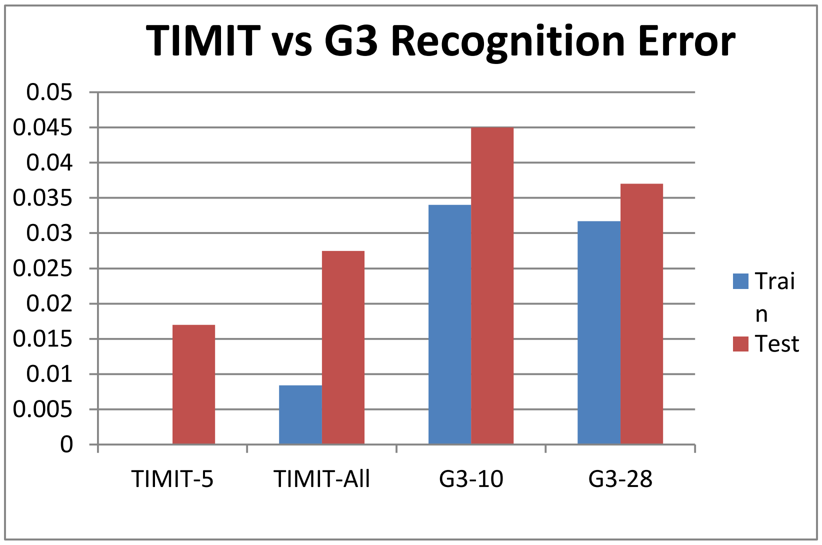

Figure 1: TIMIT vs G3 Recognition Error. The TIMIT data has lower error than the G3 data for the tasks. The TIMIT NN had 1 4-node hidden layer and used BR. The G3 data used a 10-node and 12-node hidden layers and LM

4 Experiments

We tested numerous features extraction reductions on our audio data. We inputted the feature vectors to a neural network to generate baseline scores. We then tested the same features using a convolutional neural network to determine effectiveness. In addition, we ran experiments on varied network parameters.

4.1 Feature Vector Experiments

In order to generate more audio data, we split each wav file into smaller chunks before performing reductions.

(1) Timewise Reduction: In this reduction we collapse the time axis of the spectrogram by taking the average over time of the amplitude for different frequencies. We expect variations over time in the amplitude to be more due to the exact text being said and not necessarily from interesting information about the speaker.

(2) Normalization: For each dimension of the data, we subtract the mean and divide by the standard deviation in order to achieve zero-mean and unit-variance normalized input data.

(3) Pitch: The pitch feature was estimated via the cepstral method from time domain auto-correlation coefficients of the audio signal[5].

(4) Noise: Because both the TIMIT and G3 audio data were recorded over clean channels, we wanted to explore the effect of adding noise into the data. To do this, we added Gaussian white noise to the wav files.

4.2 Network Parameter Experiments

(1) Training Function: There are two options for training function. By default, Levenberg-Marquardt (LM) is used because of its speed. However, for smaller datasets, Bayesian Regularization (BR) is preferred due to its increased accuracy.

(2) Network Architecture experiments: We experimented with changing the network architecture to see how it impacted the learning process. We tested three sizes: 1 hidden layer of 10 nodes, 2 hidden layers of 10 then 12 nodes, and 3 hidden layers of 10, 12 and 8 nodes respectively.

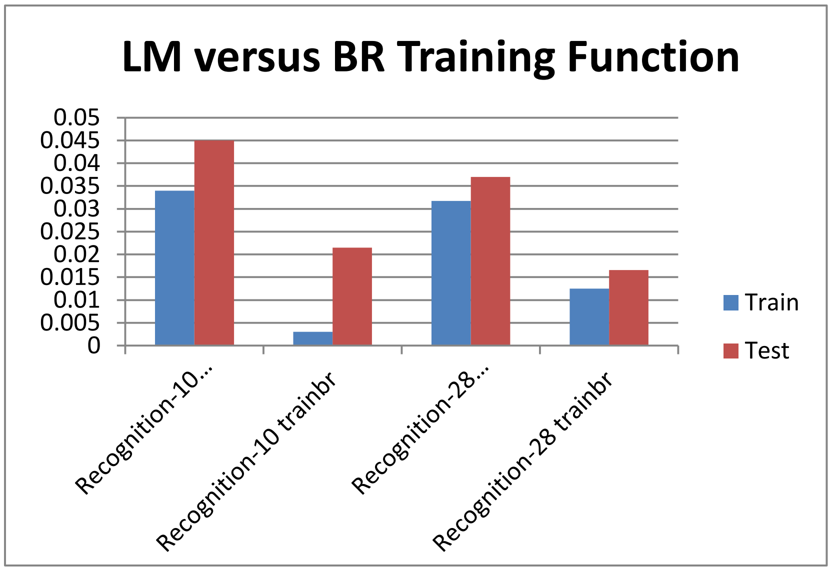

Figure 2: LM vs BR Error. While errors are low overall, BR outperforms LM with consistently lower error in the tasks. It appears that the G3 data does not have enough sound clips per speaker to make LM have lower error than BR. We prefer to use LM due to its much shorter training time

5 Results

5.1 TIMIT versus G3 Corpus

TIMIT recognition was tested on a simple network with a single 4-node hidden layer using the BR training function. G3 corpus was trained using a 10, 12-node hidden layer network using the LM training function. The resulting train and test error are shown in Figure 1.

We found that even though the network for TIMIT was smaller, the classification error was lower. This is because the G3 corpus is more difficult to classify since it contains no distinct variations in dialect, all male voices, and because the audio clip speech is more diverse.

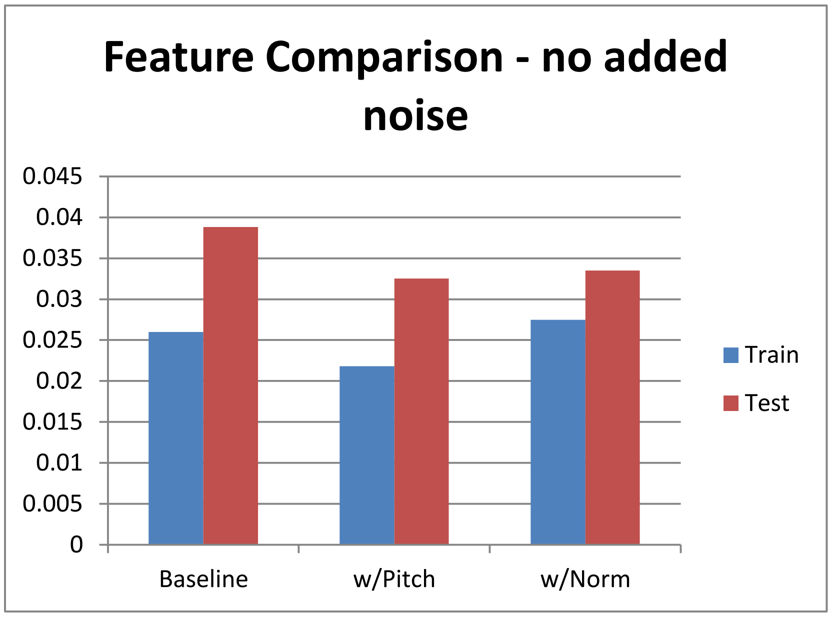

Figure 3: Recognition-10 on Feature Experiments with no Gaussian Noise added. Baseline uses the time-averaged amplitudes of each frequency of the data's spectrogram. The pitch (middle) found by autocorrelation in the time domain shows a possible improvement in the test data confirmed in Figure 4. Normalization to mean-0, variance-1 unexpectedly does not improve results, as verified in Figure 4.

5.2 Network Parameters

We compared LM and BR on two different recognition tasks: between 10 people and between 28 people (the largest set of people that matlab's memory could handle). We found that BR was more accurate but LM was faster in both tests. This confirms our previous knowledge about LM and BR. The accuracy comparisons are shown in Figure 2.

5.3 Feature Results

When testing, we gave the networks input data and specified a 70-30 train and test split. We found that our feature experiments generally did not improve on the baseline that was already very accurate (Figure 3). Surprisingly, normalizing the input data using zero-mean, one-variance did not improve results. We believe this is because we used the LM training function that is invariant to scaling. Pitch was the only feature that improved the classification. Adding Gaussian white noise into the data decreased accuracy and verified these results (Figure 4). When both the clean and noisy data were normalized, training time was consistently lower than when left unnormalized without a noticeable difference in performance (Table 1).

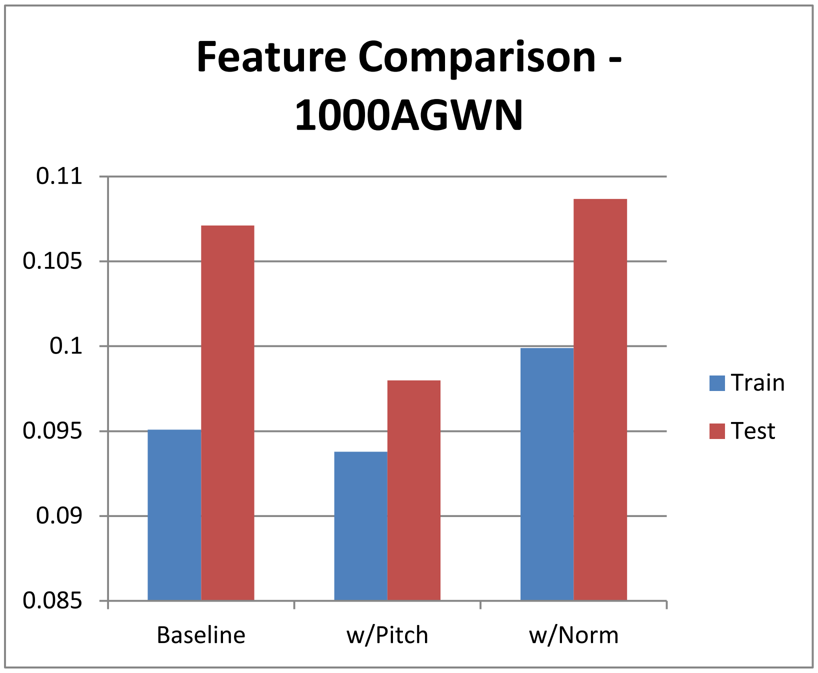

Figure 4: Recognition-10 on Feature Experiments with 1000 Additive Gaussian White Noise. The noise increases the error rates, making it clear that adding Pitch reduces the error while normalizing the data does not.

Since the accuracy of the neural network fluctuated with the number of layers, we were unable to draw conclusions about the benefit of larger networks. This inconsistency arises from the simplicity of the problem. The one layer network had accuracy of over 95% so adding more layers to the network does not necessarily increase performance for this problem.

5.4 NN vs. CNN

We found that CNN consistently performs worse than the regular NN in both speaker verification and recognition. For verification, we took created a universal background model by taking a few audio clips from each person in the corpus and attributing them to new person, UBM. We found the high errors produced by CNN interesting and experimented by running the original digit recognition data that came with the CNN implementation through the NN. We found that the NN performed better on the digit data as well. Thus, we conclude that the CNN was poorly implemented.

Providing vector instead of matrix input to the CNN provided similar results. It is probable that once again our speaker tasks and data are not complex enough to benefit from the use of CNNs.

6 Conclusion and Future Work

6.1 Conclusion

Because the baseline recognition and verification system was already very accurate, individual features did not tend to improve the performance. Additionally, CNNs seem to perform worse than NN at speech tasks in general. However, this decreased performance may be the result of using a bad implementation of CNN or having data that is so clean that preprocessing has little effect.

6.2 Future Work

Future work in this area should focus on combining and exploring new features to represent the input data. It is possible that where individual features fail to improve performance, a more optimal combination of features would help.

Additionally, future work should try to use both the female and male audio in order to test the non-dedicated imposter speaker verification tests. Since we are only using male audio, we assumed a dedicated imposter for speaker verification. Finally, we hope to experiment with data collected from noisier channels. It is possible that our clean data and AGWN data was difficult to improve upon using these features. Thus, we hope to run experiments on lower quality signals and see if our features improve performance.

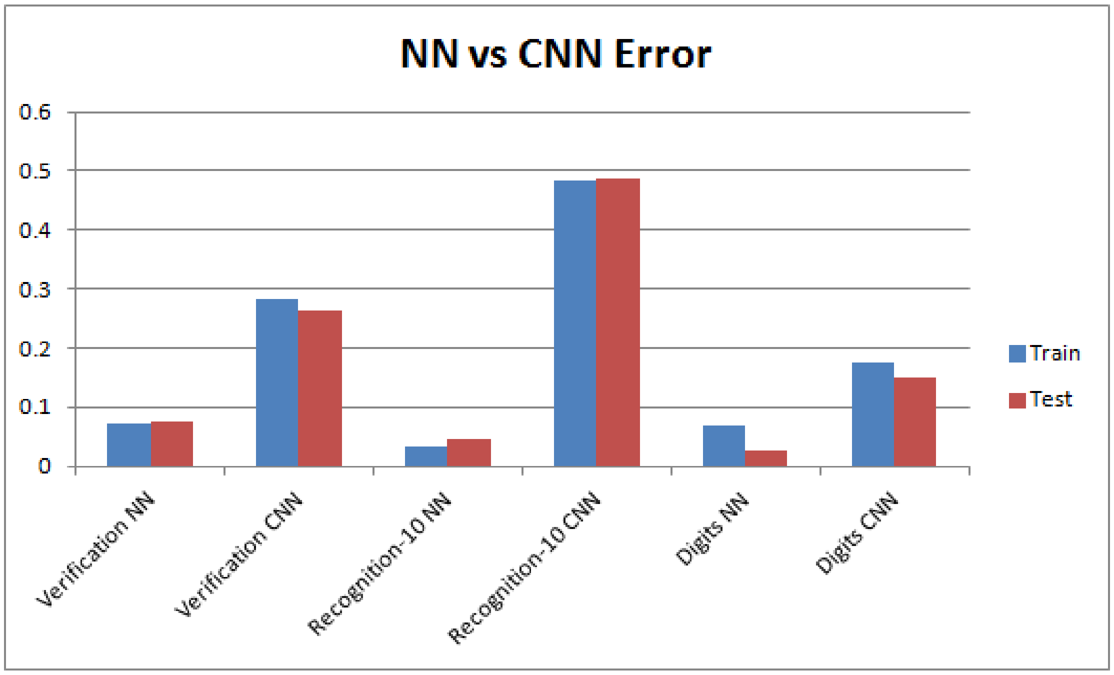

Figure 5: NN vs CNN Error. The CNN consistently performs worse than the NN. The errors in the Verification and Recognition-10 tasks were extremely high compared to the NN baseline. The original CNN implementation was intended for digit recognition, but NN still outperforms CNN on that data set. We conclude that we used a poor CNN implementation.

6 References

[1] H. Lee, Y. Largman, P. Pham, and A. Y. Ng. Unsupervised Feature Learning for Audio Classification using Convolutional Deep Belief Networks. *Advances in Neural Information Processing Systems (NIPS)* 22: 1096-1104, 2009.
[2] D. A. Reynolds. Automatic Speaker Recognition Using Gaussian Mixture Speaker Models. *The Lincoln Laboratory Journal* 8.2: 173-192, 1995.
[3] T. Kinnunen and H. Li. An Overview of Text-Independent Speaker Recognition: From Features to Supervectors. *Speech Communication* 52.1: 12-40, 2010.
[4] M. Skowronski, *WAVREADTIMIT.M* [source code], April 15, 2004. http://www.cnel.ufl.edu/~markskow/software/wavReadTimit.m, November 10, 2012.
[5] N. Seo, SPCORRELUM.PY[source code], April 2008. http://note.sonots.com/SciSoftware/Pitch.html, December 11, 2012.