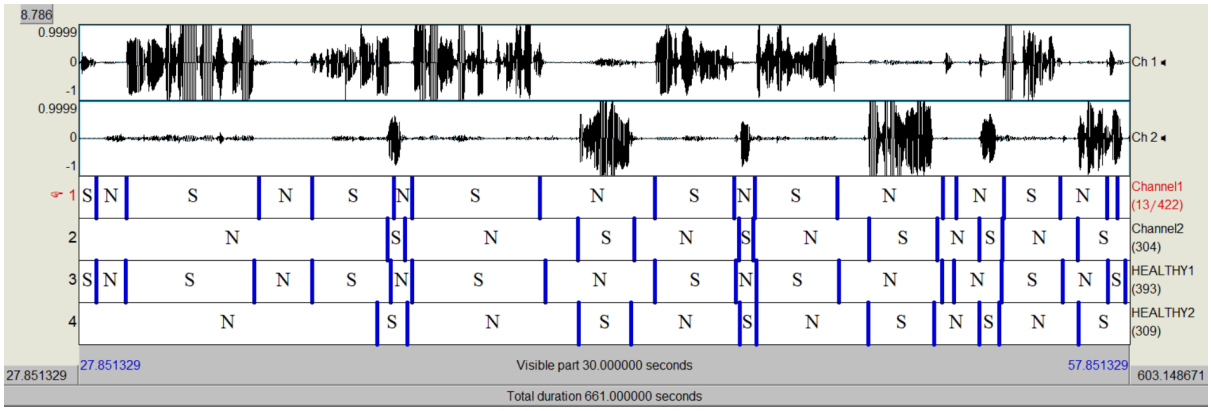
Neural Networks Diarization Project

David Lopez, Erik Sargent, Joshua Smith

December 13, 2017



1 Abstract

Speech diarization is the process of determining, in a conversation with multiple speakers, when each speaker is or isn't speaking. It is useful in speaker recognition, speech to text, clinical studies of speech impediments, etc. The purpose of this paper is to report on the development, testing, results, and usage of a convolutional neural network designed to automate the process of speech diarization.

The neural network is extremely successful in performing speech diarization and achieved a 95.05% accuracy over the entire dataset given as a sample.

Contents

2 Pre-Neural Net Processing

The provided audio files had a sample rate of $44.1kHz$, which is about 10 times what the human voice can produce. Loading all of the audio files into a neural network very quickly proved to be very memory restrictive, using about 20GB of RAM. Because of this, all of the audio files were first run through a low-pass filter and downsampled. Python struggled to downsample the large audio files, so they were instead run through a simple Matlab script functioning as a low-pass filter at $1kHz$, and then downsampling to $4.41kHz$. The downsampled data only requires $400MB$, rather than $20GB$, which makes training substantially faster. There is a slight difference when listening to the downsampled and pre-downsampled audio files, however, it has not shown to have a negative impact on the performance of the network. The Matlab script will be provided.

Each audio sample is a signed, 16-bit integer. This results in integer values from -32768 to 32767. Our output values are floating point numbers between 0 and 1. The network was somewhat successful at handling the differences in ranges. However, normalizing the audio waveforms to floating point numbers between -1 and 1 made a dramatic improvement in the performance of the network (from 60% correct to 90% correct).

The data size was also further reduced by only taking 500 ms chunks of data to input to the neural network. The neural network then classifies each 50 ms chunk of the input. This greatly reduces the data processed while still maintaining a resolution of 50 ms.

3 Architecture

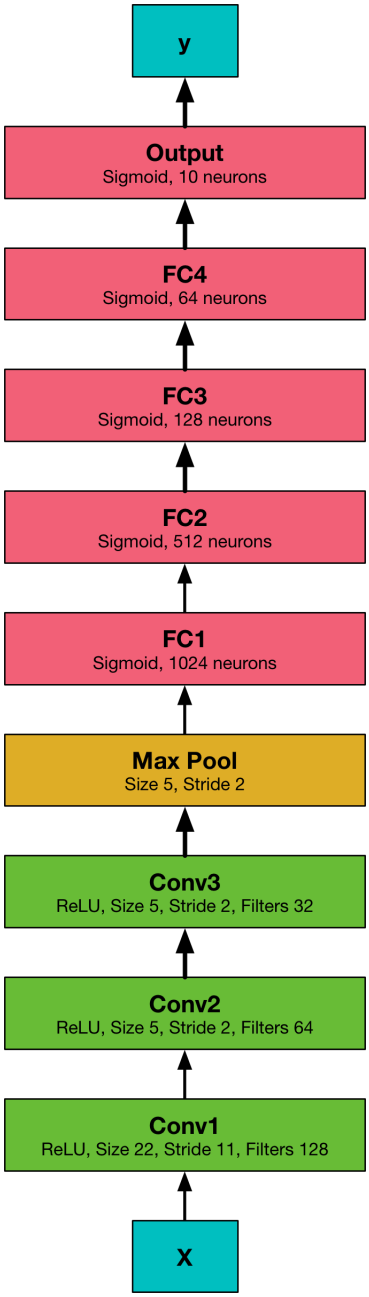


Figure 1. Structure of the neural network.

The input to the network is a tensor representing $500ms$ of data. The CSV classification data is processed and sampled as a boolean value every $50ms$. For the $500ms$ of input data, there will be 10 points that describe the speech utterance. This makes it really easy to train the network to produce a similar output which can easily be converted back to start and stop time values. An example of the CSV data chunking process is shown in Figure 2. This format for handling the input and output data greatly simplifies the data processing, and makes analyzing and learning much easier.

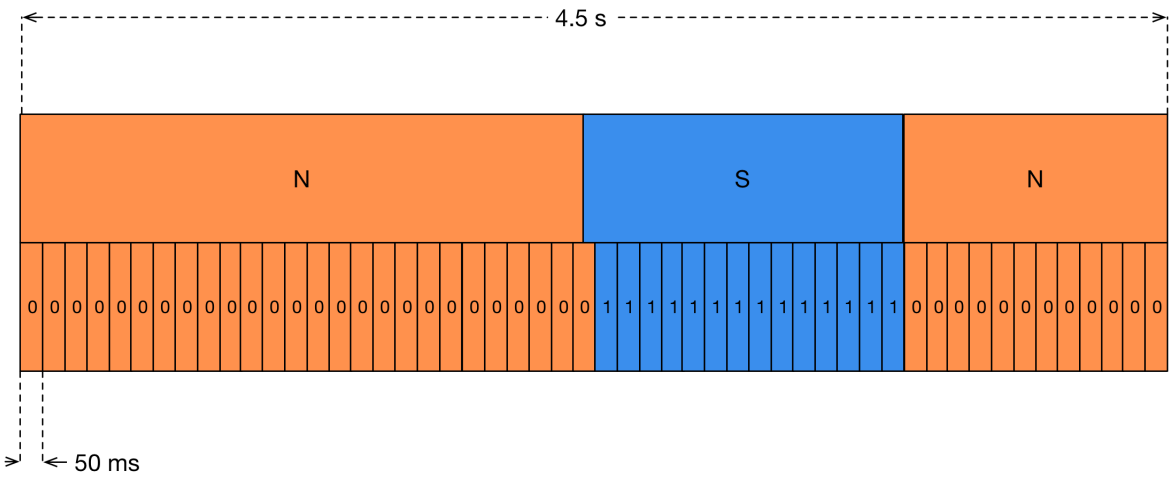


Figure 2. Example of the chunking of the CSV data.

The convolutional neural network architecture used to perform the diarization contains nine layers as seen in Figure 1. The first three layers are convolutional and function to extract increasingly large features in the sound data. The kernel sizes used in these layers are significant in that they represent approximately the same amount of data containing features of human speech that happen on the order of 20 to 50 ms. The stride was chosen for each of the convolutional layers such that as the kernel passes over the input to that layer it overlaps itself by 50%. The number of filters was chosen through a process of trial and error in which different values were tested and the current setup yielded the best results.

As is common with convolutional neural networks, a max pooling layer was placed at the output of the last convolutional layer to extract the most important and prominent features. The kernel size and stride were chosen to match that of its input so as to not miss or skip features extracted from previous layers.

Following the max pooling layer are four fully connected layers with a steadily decreasing number of neurons in each. The steady decline of neurons was chosen in an attempt to eliminate bottlenecks associated with compression of data. The prominent features extracted by the max pooling layer are gently processed and compressed into the final output layer of ten significant neurons. Those 10 output neurons each represent a $50ms$ classification window.

4 Training

A set of 37 sound files and 74 csv files were provided as training material for the neural network. Each sound file contained a conversation of approximately 15 minutes between two people. Two csv files (one per speaker) were associated with each sound file specifying the intervals over which each speaker was either vocal or silent.

The sound files and csv files were organized and processed as described above then split into two sections of 80% and 20%. The larger section was labeled as the training set and the smaller used to test the network.

Training was performed in batches of size 50 meaning that 50 pairs of 500 ms sound data chunks and their associated expected outputs were chosen at random and then presented to the neural network and then updates were made to the kernels, weights, and biases in the neurons. The updates were performed by TensorFlow's `AdamOptimizer` with default arguments. That optimizer was chosen because it performed the best when tested against `GradientDescentOptimizer` and `MomentumOptimizer`.

In an attempt to eliminate overfitting, techniques such as dropout and early stopping were included in the training process. The neural network was trained with 34 iterations over the training dataset and took approximately one hour and fifteen minutes to train. The final peak accuracy of the neural network in performing diarization on the **testing** dataset was 94.02%.

5 Results

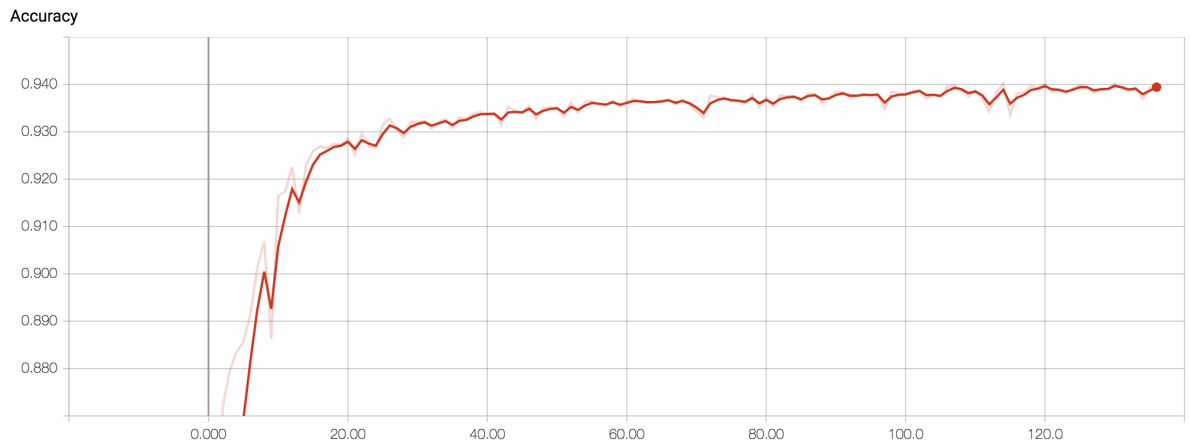


Figure 3. Learning Curve of the CNN (% Accuracy on testing data)

The peak accuracy achieved by the neural network on the **testing** dataset was 94.02% as seen in Figure 3. The accuracy on the **training** dataset was also measured and peaked at nearly 96%.

As an additional measure of accuracy, the entire dataset, training and testing, was run through the network and compared bit by bit with the expected output. This test concluded an accuracy of 95.05% which is extremely desirable.

We also developed an interface to the neural network that accepts an input and output directory, reads in all the downsampled sound files from the input, and outputs TextGrid files that can be read by Praat containing the diarization data for each sound file. An example of this generated data can be seen in Figure 4 side by side with the given labeled data. As can be observed, the diarization is extremely accurate.

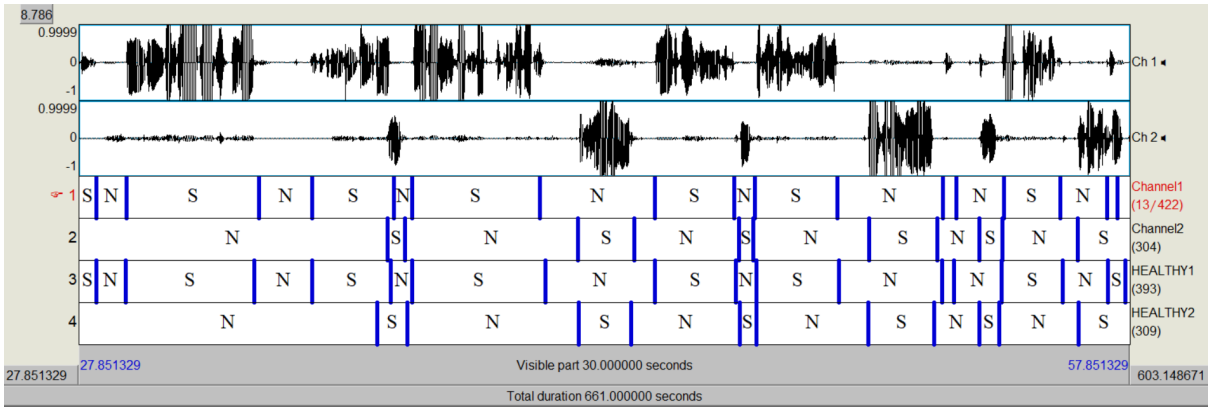


Figure 4. CNN Generated Data (Channel) vs Original Praat Data (Healthy).

While reviewing differences between the provided labeled data and the data generated from our network, we would occasionally find pieces of audio that our network marked as non-speech, but the labeled data listed as speech. Figure 5 shows one of these cases. The highlighted non-speech block is actually the person inhaling after speaking, and then they continue speaking afterwards. This section was actually mis-labeled by the person classifying the data, and the network did classify it correctly as non-speech. The rules for an utterance state that a break in speech must be at least half a second, and the pause is 0.65 seconds.

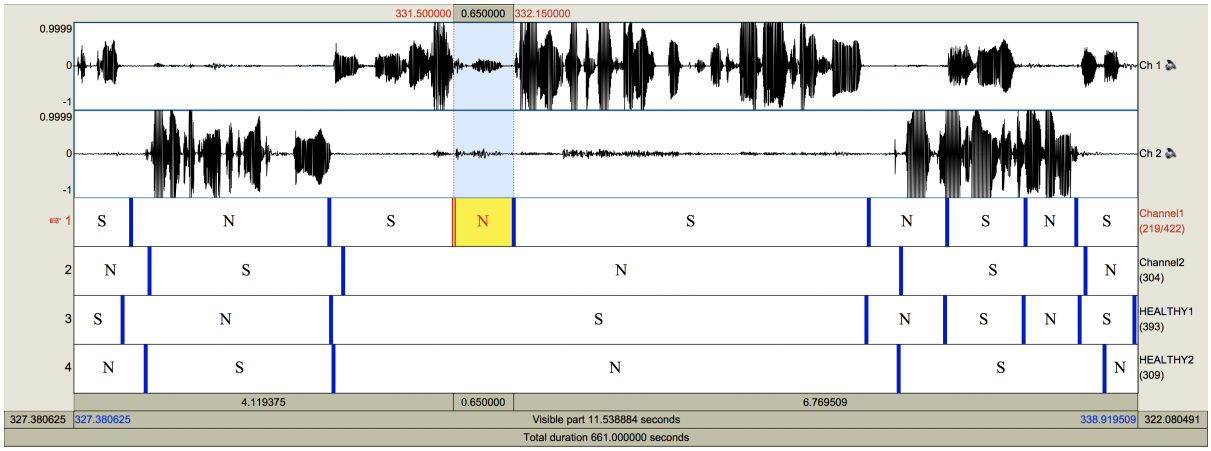


Figure 5. Demonstration of an error from the sample data (breathing noise that we correctly filtered out).

Comparing the generated classifications with the provided waveforms we have identified several similar places per audio file where the network classified the data correctly, but the labeled data was wrong. However, there are also a few places where valid speech was classified by the network as non-speech. There are a few places we have noticed that a speaker either moves away from the microphone or talks really quietly, so waveform audio looks like they are not speaking. Listening to the audio, however, shows that they are actually talking, but very quietly. The human labeling process correctly identified this as speech, but our network did not.

6 Other Methods/ Items Tweaked

It was noticed that there is a small degree of crosstalk between the two speakers. We considered providing the autocorrelation between the two speakers to attempt to clear the crosstalk. The idea being that the autocorrelated data could provide insight into the relationship between the two speakers for the neural net instead of just letting it find the relationship on its own. We ended up not building this method because the cross talk between the channels does not appear to cause problems with learning and detection.

It was also noticed that the audio files have lots of breathing sounds (or other similar noises) mixed in with the speech. We considered finding a method to filter out the breathing noises or training a neural network to recognize and automatically filter out the breathing sounds. However, this would require having a labeled dataset of the breathing noises to train on, which was not available. The network was actually able to successfully filter out the breathing noises on its own, without any additional help from us.

Within the neural net, some of the values we changed were the number of layers, neurons, kernel size, etc. Due to the nature of neural networks, most of these decisions were mostly through guess work and chosen based on the result. The net currently uses nine layers as there is a plateau after nine. A Sigmoid function was chosen as the activation function for the fully connected layers after various tests. A network with 6 layers (2 convolutional, 1 max pooling, 3 fully-connected) performed almost as well (93% on the testing data), and was able to train in considerably less time than the 9 layer network. A smaller network was much more practical for testing theories and changes quickly, and the larger network was better for final accuracy and network training. The smaller network can be reasonably trained in 10-20 minutes.

"The definition of a spoken utterance (S) is a single speech utterance, including pauses of up to 0.5 seconds." The neural network does not have a strict way of knowing this rule, so the data was swept through to replace moments of "non-speech" that were shorter than 0.5 seconds to the previous decision. This actually reduced the performance of the system so it was removed. The network learned the strict utterance rules well enough on its own to perform really well.

7 What We Learned

One of the most important lessons we learned was to not discount the effect seemingly small changes can have on the neural network's learning pattern. One change that caused over 40% difference in performance was to normalize the data. Another change was to use a ReLU Function for the convolutional layer as opposed to Sigmoid. This yielded a 30% difference. These were not obvious or expected changes that could cause such a difference. While many things can be intuitive about a neural net, they are still alien enough that it can be worth turning every rock.

Another lesson we learned was to study and understand our training data so that we can find reasons the neural net might have disagreed with it. There were many instances in this project in which we could side with the decision of the neural network as opposed to the training data. The training data may not always be without fault and that is cer-

tainly the case in this project. It is also interesting to note that on two fronts our neural net would be more accurate if the training data were more accurate. This is because an answer could change from the training data that would suddenly make our neural network's decision correct. More importantly, the neural network would have better data to learn from to reinforce or improve some of the decision it made.

A portion of the project that took a lot of time was the data organization. Whether it was downsampling, reducing, or setting up the form in which the neural network would accept the data, the data format was very important to the project. Understanding the type of neural network that we are using also gave key insight into the best way to present the data. Taking the time to think about these details contributed greatly to the success of the neural network's learning.

8 Usage

The code developed for the project is included below, but can also be downloaded from `https://github.com/eriksargent/nn-diarization`. It is important to downsample the sound files as described above using the Matlab file `ds.m`. The project has the following dependencies: `Python 3.6, TensorFlow, NumPy, SciPy, Matlab`. Once the environment is set up, the following commands can be used to manipulate/run the diarization module.

8.1 Processing the input data

```
python3 read_data.py /path/to/downsampled/data
```

The `read_data` command will handle all data conversion and will create a file called `data_cache.npz`. This is a numpy cache of the data that was generated. This command only needs to be run once, and the training of the network will always read from the generated cache file.

8.2 Training the CNN

```
python3 cnn9layers.py
```

This terminal command is used to train the convolutional neural network. This command, depending on the hardware that the program is running on, will likely take at least an hour.

8.3 Running the diarization module

The git repository contains a python module called `diarization.py`. This module is a wrapper that references the last trained CNN and takes care of reading in the sound files it is pointed at, passing them to and collecting them from the CNN, and outputting the generated data in the form of Praat TextGrid files.

```
python3 diarization.py /path/to/downsampled/data /path/to/output/dir
```