Nate Johnson
Brinkerhoff, CSCI 547
Final Project, F2019

Neural Network Classification of Non-native English Accents

Data

The data used is from L2-ARTIC, an extensive and carefully curated collection of recordings of non-native English speakers. There are speakers of six languages represented: Arabic, Mandarin, Hindi, Korean, Spanish and Vietnamese. There two men and two women speakers of each language, so 24 speakers in total. Each speaker read the same 1132 prompts, which are all presented as separate audio files. All in all, it is about 27 hours of audio (more than an hour per speaker).

RNN

The audio genre classification project presented in class seemed very similar to this task. My research began by seeing if an RNN on this dataset would outperform the CNN presented in class. See the attached file RNN Audio Classifier for this. I tried different criterion and optimizers, but none of them exhibited much learning from epoch to epoch. Oddly, the test set also outperformed the training set (training 48%, test: 57%) I concluded I don't know how to design such a network for this task, and moved back to the CNN. I include it here for the record, perhaps it will yield to analysis later.

CNN1 Initial Data Processing

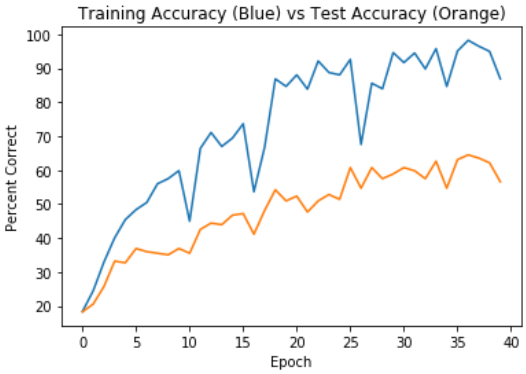
File length for the same prompt varied a lot. Usually above 15%, and sometimes as much as 24%. The recordings are very short as well (averaging 3.6 seconds), and sometimes not even a second. The initial design called for fixed size inputs. An arbitrary decision was made to use a file representing about 12 seconds of speech. This was chosen to for convenience regarding file size and to create an utterance with some length so that features such as changing intonation might be clearly presented. In order to guarantee a minimum length of 12 seconds, files were concatenated 25 at a time and then truncated. The files had been sampled at 44100 Hz and were down-sampled by a factor of 8 to reduce file size to 65536 features.

CNN1 Net

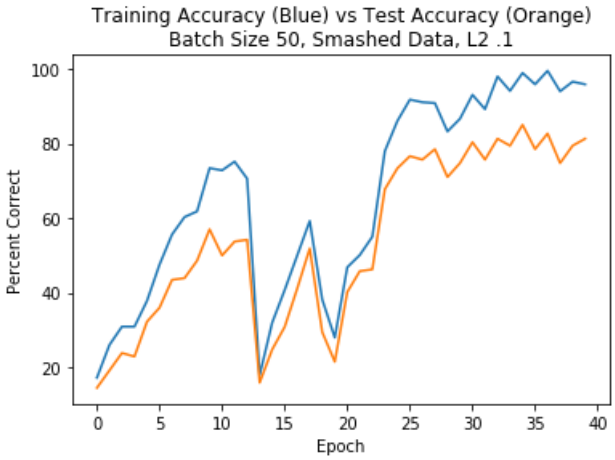
I used the same structure as presented in class.

CNN1 Net Results.

The model suffered from overtraining as the practice set approached 99% accuracy while the training set was around 60%.



To mitigate this, L2 regularization was tried by setting the weight_decay parameter on the Adam optimizer to 0.1 The best run on these settings produced a training set accuracy of 98.9% and a test set accuracy of 86%. The results varied between 76% to 86%, with a typical run being 81%.



CNN2

Since all speakers were saying the same thing, it would be meaningful to somehow base analysis on matching prompts from different speakers. A search on neural networks for paired data turned up an idea used in a Google project called FaceNet.[i] Two images representing the same person were paired with 1 different person. These were encoded using the one-hot method. The papers claims the method improved accuracy by 30% over the best known published results.

For our purposes, we group 3 speakers all saying the same utterance, two from the same native-language family and one from a different family. With six languages, choosing 2 gives 15 possible combinations. Since for any two speakers a and b, the triplet could be aab or bba, the number of combinations increases to 30. With 4 speakers of each language, there are 4 * 3 * 4 ways to satisfy any combination, given 1440 speaker combinations with this data set. Since each combination uses 1 prompt, 1440 times the number of prompts generates the number of input records.

The advantage of this approach is two-fold. First, we are comparing the same utterances between different speakers. Second, we can dramatically increase the number of input records. The first method, based on concatenation, created only about 1120 files (albeit with longer feature length per file). This method starts at 1140 times the number of prompts, and so could scale to over 1.5 million files. The disadvantage is that each record is a different length. This required some method to make the feature lengths match, at least for the 3 files that are grouped together for comparison. As a first attempt, the shorter of the three were zero padded out to match the longest. Each group of three can still vary though, so this method can't support batch processing

Unfortunately, I was unable to design a neural network to implement this. Since image processing with color images uses a 3D input vector, that did not seems to be the problem. However, I could not get the CNN to run with different length inputs. If I had solved that problem, the next problem would have been handing the classification encoding. Each input was encoded as a list of integers, e.g. [0,0,1] or [2,2,4], which represents two speakers of class zero and 1 of class 1, etc. I could change them to one-hot, but it would have to be a 12 element vector- 1 position if there are 2 of that type of speaker, and another position if there is only 1 of that type of speaker, times 6 types of speakers = 12 slots. I anticipate that would have been difficult to work with in.

In conclusion, this was an interesting project. I think the second method that I was unable to implement has a lot of promise. It is challenging working with data this size. I have serious doubts how well this would transfer to recordings of speakers who weren't in the original data set, but it would be very interesting to try that.

---

[i] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering." ArXiv.org (2015): ArXiv.org, Jun 17, 2015. Web.