Forced-alignment of the Sung Acoustic Signal Using Deep Neural Nets

Dallin Backstrom[*1], **Matthew C. Kelley**[†1], **and Benjamin V. Tucker**[‡1]
[1]Department of Linguistics, University of Alberta

1 Introduction

Sung speech shows significant acoustic differences from spoken speech. One challenge in analyzing both spoken and sung speech is identifying the individual speech sounds. Forced-alignment systems such as P2FA [1] and the Montreal Forced Aligner [2] have been designed to accomplish this task for spoken speech, however, there is no such tool for sung speech. Previous work used a combination of hidden Markov models and convolutional neural networks on log-Mel filterbanks to segment phones in sung Mandarin opera [3]. We, in turn, trained a deep neural network to extract phone-level information from a sung acoustic signal. The primary objective was to create a model that can take a WAV file containing a target song as the input, and produce time-aligned phonemic labels automatically as output. To measure the performance of our model on these tasks, we primarily measured accuracy on identifying the correct phone label at a given time-step. We also compared the accuracy of our model to other state of the art systems, trained on spoken speech, performing the same task with sung speech.

2 Method

We used a selection of traditional Canadian folk songs from the Moses and Frances Asch Collection [4] to train our model. We selected songs from the collection that had either no or minimal accompaniment. There were two unique singers in the data set, one male [5] and one female [6].

2.1 Building and Training the Model

Time-aligned labels for the data were created by manually transcribing the songs using a Praat TextGrid [7]. Approximately 30 minutes worth of songs were transcribed in total. Twenty-five (25) millisecond windows of audio spaced 10 milliseconds apart were then extracted from the recordings. The label for which phone class the window belonged to was determined based on the time-aligned transcription. To increase the amount of data a second copy of each window was created by adding Gaussian noise to the audio and then extracting the windows and labels again. We also used the TIMIT [8] spoken speech data set.

Using the Keras deep learning library with the Tensorflow backend, we built several models to train on our data. Our most successful model had an architecture comprised of two convolutional layers, four bidirectional LSTM layers, and a time-distributed result layer. The first convolutional layer has a filter size of 4 and 512 filters with a stride length of 1. The second convolutional layer has 256 filters and a filter size of 4 with a stride length of 1 and a dilation rate of 2. Both convolutional layers used Keras's "causal" padding option. The convolutional layers are both immediately followed by a spatialized dropout layer at a rate of 30% and a batch normalization layer with default parameters. The first, second and fourth Bidirectional LSTM layers had a size of 256, while the third had a size of 512. Default batch normalization was again used and the LSTM layers all had Gaussian dropout at a rate of 60%. All other parameters were left at their default values.

The model was trained with categorical cross-entropy loss. Its framewise phone recognition accuracy was monitored during training to identify the best performing model. The Adam optimizer was used, with the default configuration. This model was trained first for 10 epochs on our singing dataset, then for 10 epochs on the TIMIT corpus of spoken speech, and then was trained again on our singing dataset for another 10 epochs. The batch size for each of these training routines was 1. By training on TIMIT, the model was exposed to a far greater amount of data than it would otherwise have seen, and because the TIMIT data is formatted in the same way as the singing data, the model is more robust as a result.

3 Results

Table 1: Accuracy of Different Model Architectures

Model Architecture	Highest Accuracy
Convoluional	18 %
Convolutional + LSTM	53 %
Convolutional + Bidirectional LSTM	81 %

Our best performing model achieved training accuracies of approximately 80 percent (see Table 1). However, ultimately the output that we want to achieve is not a list of the most probable phone every 10ms; rather, we want to know where the boundaries fall between each phone. To do so, we use the decoding algorithm specified by Kelley & Tucker [9], but using standard backtracking instead of the argmax routine, to produce time stamps for the boundaries of each phone. Of course, if the model had correctly labeled every phone with 100% confidence at each time-step, this task would be trivial, which is why we used accuracy as a training and evaluation metric.

Figure 1 presents a sample alignment of a short segment of a song. Analysis of the quality of the automatic transcriptions overall is more difficult, but as demonstrated in Figure 1, vowels are segmented acceptably well, and certain other segments are also consistent, such as the burst releases of /t/

*dbackstr@ualberta.ca
†mckelley@ualberta.ca
‡benjamin.tucker@ualberta.ca

and /d/ and sonorants like /ɹ/. In fact, the model tends to overestimate its confidence in these segments and compress all of the phones that it is not as confident about into a tiny margin in between the ones that it is more confident in. This is an undesirable behaviour, but we encountered this same tendency, to minimize certain phones in models used on this task, for regular speech. Beyond this problem, the model does place certain boundaries in nearly the same position as the hand-alignment, which is encouraging, as placing boundaries with high accuracy is necessary for the model to be useful in automating the alignment task.

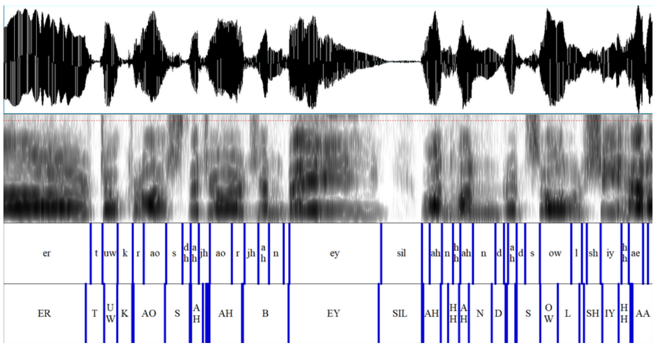

Figure 1: Sample alignment of a section of a song. From top to bottom: Waveform, spectrogram, target transcription, model transcription.

4 Discussion

One important question that arises from our results is why, despite achieving good frame-wise accuracy rates in the training set, the alignments produced by the model are so bad in certain places. Manual inspection of the alignment suggests that the model seems to almost ignore certain phones entirely, which is most likely due to it not having as much confidence in being able to identify those phones correctly when compared to sonorants and bust releases. If its confidence is spread between many classes, it will have a hard time deciding which one it should select, and may ignore those lower probability phones in favour of items on which the model reaches a much higher level of confidence. For this reason it may be very useful for us to look at the way that we translate the labels outputted by the model into a Praat TextGrid, as there might be a way to do this that takes into consideration the fact that some classes of phone will have overall lower levels of probability than others due to them appearing more similar to other options. Minimum and maximum duration constraints, for example, on the alignment algorithm should help ameliorate this behavior.

Future work should focus on determining why the model is making classification mistakes. A confusion matrix or inverse layer maximization may help indicate what kinds of mistakes the network is making.

5 Conclusion

In this project, we achieved our main initial goals of creating a model that is able to classify phones when presented with singing as well as produce a time-aligned Praat TextGrid that can be compared to the original audio track. Some of our other goals, such as using various audio pre-processing methods to compare their effectiveness, we did not achieve. Despite the difficulty in creating from scratch a dataset that would be sufficiently large to adequately train a model for this challenging task, we were able to create a model that is able to classify phones with some degree of accuracy. Excitingly, this suggests to us that this task, although more difficult than the same task performed on regular speech, can be handled in a similar way and with similar levels of success.

Ultimately, we hope to improve the model and the automatic phone alignment further, until it can be packaged as a bespoke application for use in research. We also plan to explore the many questions about how our model can classify sung phonemes, and what differences and similarities it might have with humans, as we continue to test and improve upon it.

Acknowledgments

This project was made possible through an Undergraduate Research Grant from Sound Studies Institute at the University of Alberta, and their stewardship of the Moses and Francis Asch collection of folk music. We also acknowledge the support of the NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research. The authors would also like to thank the members of the Alberta Phonetics Laboratory for their support and suggestions.

References

[1] Jiahong Yuan and Mark Liberman. Speaker identification on the SCOTUS corpus. *Journal of the Acoustical Society of America*, 123(5):3878, 2008.

[2] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal Forced Aligner: Trainable text-speech alignment using Kaldi. In *Interspeech 2017*, pages 498–502, 2017.

[3] Rong Gong and Xavier Serra. Singing voice phoneme segmentation by hierarchically inferring syllable and phoneme onset positions. In *Proc. Interspeech 2018*, pages 716–720, 2018.

[4] ”Moses and Frances Asch Collection”. Sound Studies Institute: University of Alberta, May 2018.

[5] O.J.Abbott; Artist; Edith Fawke; Lining Notes. Irish and British Folk Songs from the Ottawa Valley, 1961.

[6] Karen James; Artist; Samuel Gesser; Lining Notes. Karen James, 1961.

[7] Paul Boersma and David Weenink. Praat, a system for doing phonetics by computer, 2018.

[8] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. DARPA TIMIT acoustic-phonetic continuous speech corpus CD-ROM. NIST speech disc 1-1.1. *NASA STI/Recon technical report n*, 93, 1993.

[9] Matthew C. Kelley and Benjamin V. Tucker. A comparison of input types to a deep neural network-based forced aligner. In *Proc. Interspeech 2018*, pages 1205–1209, 2018.