

NEURAL SPEECH-TO-SPEECH SYNTHESIS

Jared Samet - UNI: jss2272

jss2272@columbia.edu

ABSTRACT

We present a system that extracts prosody features from a multi-speaker dataset. The prosody features are computed using by fitting low-degree Legendre series to the pitch and power contours of the audio for each vowel in the input audio. We cluster the features using unsupervised learning to produce a discrete set of categories for the pronunciation of English vowels. By adding the cluster labels as annotation to text, a text-to-speech synthesis system can learn to incorporate the prosody into its output. We then annotate a single-speaker speech dataset with these cluster labels and then use to train a Tacotron model. The resulting system can produce audio that is different even when the text is the same. Prosody features from utterances by new speakers can be extracted in the same way and the resulting features used to cluster and annotate the text of the new utterance. The trained speech synthesis can use the results to produce new synthesized audio that mimics the prosody from the original speaker.

Index Terms— Prosody, unsupervised learning, speech synthesis, seq2seq

1. INTRODUCTION

I used Kaldi [1] to create an alignment for the Tedlium data using the final triphone model it created. For each vowel phoneme, I used Kaldi's pitch extractor and the first MFCC component (energy) to create two series of numbers. Kaldi's pitch extractor is already normalized but I used [StandardScaler] to normalize the energy component across the utterance. I fit a second-degree (?) Legendre polynomial to the pitch and power to create six features for each vowel. The duration gave me the seventh feature. I then ran K-means clustering on these to create eight different vowel clusters that were common across the entire range of speakers in the (sub-sampled) Tedlium data.

I then used the same triphone model to generate an alignment for the LJSpeech dataset, extracted the same pitch, power, and duration features for LJ, and used the previously computed vowel clusters to assign a cluster label to each vowel in the LJSpeech dataset. I (slightly) modified the Tacotron implementation to accept a sequence of tokens instead of a sequence of characters. Instead of text characters, my input tokens consisted of (Kaldi's) phonemes and the

vowel cluster labels. Having suitably modified Tacotron, I then trained Tacotron on the [phoneme + cluster label, audio] pairs.

Finally, to see if it worked, I recorded myself saying a sentence in multiple ways, ran each .wav through the same align + label steps, and fed the resulting [phoneme+label, text] pairs to Tacotron. She said the same thing different ways.

2. RELATED WORK

This project involved two main components: first, extracting prosody features from a set of input audio files; and second, training a text-to-speech synthesis model on a dataset that had been labeled using the extracted prosody features.

Selkirk [2] discusses sentence prosody and pitch accent in the context of English. Although English is generally not thought of as a tonal language, Selkirk writes that “[i]n English a pitch accent associates to a stress-prominent syllable in a word (typically the main word stress.)” Ghahremani et al. [3] describes a pitch-extraction algorithm (“the Kaldi pitch tracker”) based on Talkin [4] that is specifically designed for use in the speech recognition concept and is implemented in the open-source Kaldi project [1]. This project uses that implementation to extract the pitch contour. Fujisaki [5] models the F_0 contour over the duration of an utterance as the sum of a set of impulse response and step response functions, parameterized with a finite number of scalar values. Wang et al. [6] use the pitch and amplitude contours to improve tone recognition in Mandarin by identifying “maxima, minima, and inflection points of particular acoustic events.” Wong and Siu [7] use robust regression and orthogonal polynomials to create features for a decision tree classifier in order to recognize tones in Chinese languages. Finally, Lin [8] and Mary [9] use a small number of Legendre polynomial coefficients to represent the pitch contour as a finite-dimensional feature vector, which is the approach used in this project.

Speech synthesis or text-to-speech is a well-studied problem that has been actively researched since the 1950s. While there has been remarkable progress in the field in recent years, the quality of computer-generated speech has not yet reached human levels. Current commercial systems described in Khan et al. [10] and Taylor [11] generally use concatenative speech synthesis to produce their output. However, the alternative approach of parametric synthesis using neural networks is

rapidly gaining popularity, with several papers since 2016 demonstrating impressive results in the quality of the output. The first of this generation was Google’s WaveNet (Oord et al. [12]), followed in quick succession by Deep Voice and Deep Voice 2 from Baidu (Arik et al. [13], [14]), Char2Wav from MILA (Sotelo et al. [15]), and Tacotron from Google (Wang et al. [16]). Each of these systems has taken a different approach to the network architecture to address different aspects of the speech synthesis pipeline. Tacotron, which is the back-end used in this project, is an end-to-end text-to-speech system based on the sequence-to-sequence with attention model. As the authors describe, “The model takes characters as input and outputs the corresponding raw spectrogram, which is then fed to the Griffin-Lim reconstruction algorithm to synthesize speech.”

3. OVERVIEW

The goal of this project was to create a system that, given an input audio file from an arbitrary speaker, produces a synthesized audio output of the same utterance in the voice of a second speaker, where the prosody of the output audio matches that of the input audio as closely as possible. The system implemented uses a pipeline of several processing steps in order to accomplish this. An overview of the pipeline and a diagram are presented here for context; a detailed description of each step follows.

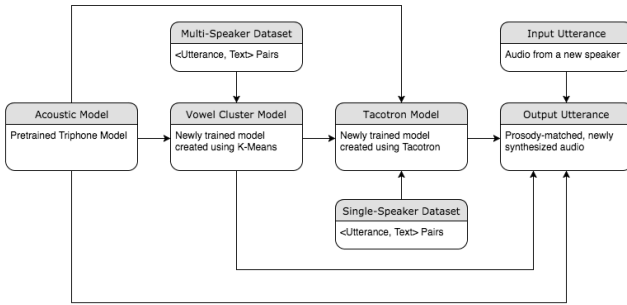


Fig. 1. Pipeline Overview

The first portion of the system is the vowel-cluster training process, which takes as input a previously-trained acoustic model for alignment and a speech dataset from multiple speakers. As output, it produces a clustering model that can be used to annotate an audio utterance from an arbitrary speaker with cluster labels for each vowel in the utterance. This portion of the system uses Kaldi to, first, perform forced alignment on the multi-speaker dataset, and, second, to extract the pitch contour and the first (energy) MFCC component for each frame of the input audio. Given the alignment, pitch, and power contours, an unsupervised clustering algorithm (K-means) trained on the audio segments corresponding

to vowels to learn several distinct ways in which syllables can be pronounced.

The second portion of the system is the speech-synthesis training process, which takes as input the pre-existing acoustic model, the newly-trained vowel-cluster model, and a large speech dataset from a single speaker. As output, it produces a trained Tacotron model that can be used to generate synthesized utterances. This portion of the system first uses Kaldi to perform forced alignment on the speech dataset and extract the pitch and power features, as before. It then uses the vowel-cluster model to produce an annotated phoneme sequence for each utterance in the single-speaker dataset. Finally, the audio and the annotated phoneme sequence pairs are used to train the Tacotron model.

The final portion of the system is generates new utterances. As input, it takes the pre-existing acoustic model, the newly-trained vowel-cluster model, and the newly-trained Tacotron model, and an input audio file in the voice of an arbitrary speaker. As output, it produces synthesized audio of the equivalent utterance where the prosody matches that of the input utterance as closely as possible. This portion of the system computes an alignment for the input utterance and extracts the pitch and power features; uses the vowel cluster model to produce an annotated phoneme sequence for the input utterance; and, finally, uses the newly-trained Tacotron model to synthesize the output audio.

4. PROSODY FEATURE EXTRACTION

All three portions of the pipeline involve extracting prosody features from the input audio – in the vowel-cluster training process, the input audio is the multi-speaker dataset; in the single-speaker annotation process, the input audio is the single-speaker dataset; and in the utterance-generation portion, the input audio is the new utterance the user wishes to re-synthesize in a new voice. The prosody feature extraction is performed in three steps; Figure 2 shows a diagram.

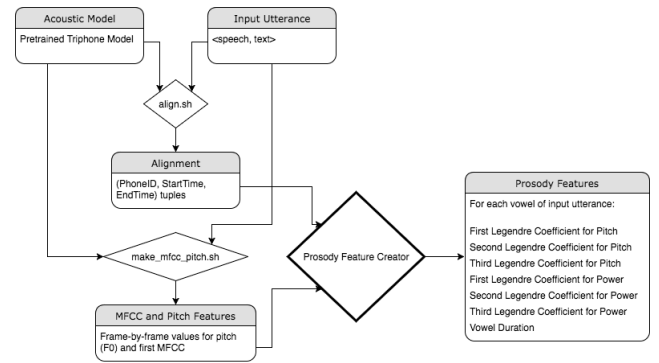


Fig. 2. Prosody Feature Extraction

First, Kaldi’s `align.sh` script uses a pretrained acoustic triphone model – in this project, the final triphone model re-

sulting from Kaldi’s TEDLIUM recipe – to compute a forced alignment of the input audio, and Kaldi’s `ali-to-phones` tool is used to convert the model-level alignment to a sequence of $(phone_id, start_time, end_time)$ tuples. Next, Kaldi’s `make_mfcc_pitch.sh` script creates the MFCC and pitch features for each frame of the input audio, the `copy-matrix` tool converts this to a text file, and my python script `kaldi_to_npz.py` converts the text file to a numpy array (.npz) file. Finally, seven real-valued features are created for each vowel phone in the alignment.

The first three features are the coefficients of the second-degree Legendre series that is the least-squares fit to a series of (x, y) points where the length of the series is the computed number of frames plus 4, the x values are evenly spaced between -1 and 1, and the y values are the frame-by-frame pitch values computed using Kaldi’s pitch-extraction algorithm, starting two frames before the beginning of the phone and ending two frames after the end of the phone. Since Kaldi’s algorithm already normalizes the pitch contour over a three-second window, no further normalization is done before computing the Legendre coefficients.

The next three features are the Legendre coefficients for the power (first MFCC component) component, calculated in the same way as for the pitch features. Since this component is not normalized by Kaldi, my system normalized the coefficient to have mean zero and unit variance over the whole utterance before computing these coefficients. The seventh feature is simply the duration of the phone.

These seven features were chosen in the hopes of maximizing the useful prosodic information available in a small set of numbers per phoneme. The two frames (20 ms) before and after the utterance are added to the sequence to improve the conditioning of the least-squares fit matrix; to compensate for potential slight errors in the alignment as computed by Kaldi; and to provide a small degree of context for the vowel in question. The domain is fixed at [-1, 1] so that the coefficients capture the level, slope and curvature over the length of the phoneme, regardless of the duration. Due to the structure of the Legendre polynomials, the three features for the pitch and power series contain information about whether the vowel is pronounced with a rising, falling, or flat tone, and whether the vowel contained a local maximum or minimum of pitch or power. Finally, the duration feature is a simple attempt to detect whether the vowel is pronounced quickly or is drawn-out.

Most importantly, these features are intended to be useful for identifying which syllables in an utterance are stressed. As Selkirk [2] discusses, “In English a pitch accent associates to a stress-prominent syllable in a word (typically the main word stress).” However, spoken English uses pitch contour to convey more information than simply which syllable is a word is stressed. Pierrehumbert [17] identifies 22 different patterns that an English phrase can exhibit, consisting of various combinations of pitch accent, phrase accent, and into-

	A(n)	O(th)	E(r)	O(r)	A(nj)
Pitch 1	-1.1/-2.4	-3.2/0.6	0.9/0.5	3.6/0.0	1.2/0.1
Pitch 2	-2.0/3.5	0.0/0.6	5.5/-0.1	-2.5/-0.2	0.8/0.8
Pitch 3	0.8/6.3	0.6/1.2	3.4/0.5	-5.0/0.5	-0.1/-1.1
Power 1	0.1/0.0	0.8/1.0	0.7/1.0	0.8/1.1	1.5/0.9
Power 2	2.7/1.5	-0.1/-0.4	-0.6/0.5	1.3/0.6	0.2/0.4
Power 3	-1.3/-0.6	0.4/-0.3	1.4/0.7	0.5/0.5	0.5/-0.4
Duration	-0.2/-0.5	0.0/0.0	2.0/1.6	0.0/0.0	-0.4/-0.4
Cluster	7 / 6	6 / 4	5 / 4	2 / 4	1 / 1

Table 1. Extracted prosody features for two pronunciations. The first number in each cell is for the tone of disbelief. The last row shows the cluster label that the vowel was assigned to.

nation occurring at the end of a phrase. These phrases carry semantic content beyond the actual words of the text. For example, different stress patterns can indicate surprise, emphasis, disbelief, or neutrality, among others.

Figure 3 displays the pitch and power contours that Kaldi computes for two pronunciations of the two-word phrase “another orange” (the phrase is from [17].) In the first audio file, the words are spoken with a tone of disbelief, i.e., “aN-OTHer orange???”. In the second pronunciation, the words are spoken with a neutral tone. The seven computed features for each vowel are displayed in Table 1.

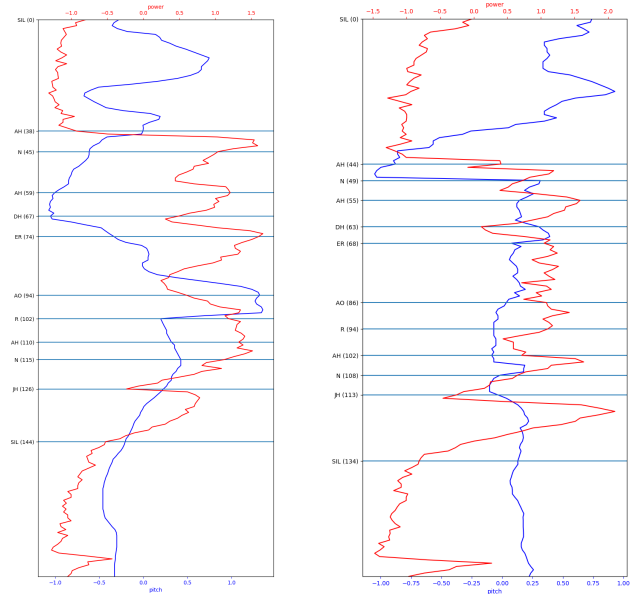


Fig. 3. Normalized pitch and power contours. The figure on the left was uttered with a tone of disbelief.

5. VOWEL CLUSTERING

To the extent that the seven extracted prosody features for each vowel do indeed convey useful information, there are numerous ways one could imagine using them to “clone” an input utterance. For example, the speech output of a speech synthesis system could be distorted in some smooth way in order to try and match the pitch and power contours of the entire input audio. Alternately, they could be used as the inputs to a supervised classification algorithm that learned to classify each syllable as stressed or unstressed, with the help of a manually-annotated dataset. The system implemented here uses k -means, an unsupervised algorithm, to partition the space of features into 8 categorical cluster labels.

As stated in Sec. 3, the input to the vowel-cluster training process consists of an acoustic model and a multi-speaker speech dataset. The prosody features for each vowel in the dataset are first computed using the acoustic model as described in Sec. 4. These features are then normalized over the entire dataset to have zero mean and unit variance on a feature-by-feature basis. Finally, k -means clustering with $k=8$ returns eight vectors in \mathbf{R}^7 corresponding to the cluster centroids. The resulting vowel cluster model therefore consists of the scaling factors that normalize each feature of the input dataset and the eight cluster centroids. This portion of the system was implemented using `scikit-learn` [18].

There were number of choices involved in how to perform the clustering, including the number of clusters, the use of an unsupervised algorithm as opposed to a supervised algorithm, and the decision to use a single set of cluster labels for all vowels, as opposed to computing clusters for each vowel individually. These choices are discussed further in Sec. 9.

6. SPEECH SYNTHESIS

The speech-synthesis training portion of the system produces a model that can be used to generate new utterances. The input to this portion of the system is the pretrained acoustic model, the vowel-clustering model resulting from Sec. 5, and a large single-speaker dataset consisting of [text, audio] pairs. In this phase, the prosody features are first extracted as described in Sec. 4. Next, the vowel-clustering model is used to assign cluster labels to each vowel in the dataset. The result is a set of [labeled-phone-sequence, audio] pairs; for a single utterance in the dataset, the labeled phone sequence for that utterance is the sequence of phones along with the cluster labels for each vowel. For example, the text for one short utterance in the dataset used here is “In 1813”. The corresponding labeled phone sequence is “IH VOWEL4 N sp SIL EY VOWEL1 T IY VOWEL4 N sp TH ER VOWEL6 T IY VOWEL2 N sp SIL”. The “sp” tokens correspond to word boundaries.

The Tacotron [16] neural network system was trained on the [labeled phone sequence, audio] pairs to produce audio.

Tacotron’s internal neural network architecture is far too complex to describe in detail in this paper. As a brief overview, it is based on the Sutskever et al [19] encoder-decoder seq2seq model with Bahdanau [20] attention. The input is a sequence of one-hot-encoded characters. The encoder embeds the characters in a continuous vector space, and passes them through the CBHG module described in the Tacotron paper. The decoder uses GRUs with attention to produce linear-scale spectrogram as the output sequence. Finally, the Griffin-Lim algorithm produces the actual audio from the spectrogram.

Since Tacotron is agnostic to the character set, the phones and the cluster labels are used here, along with the word-boundary marker, as the character set. Tacotron’s neural network architecture makes this possible out-of-the-box; it was specifically designed to accomodate end-to-end training using only audio and the corresponding text as its inputs, in contrast to the relatively complex requirements for most speech synthesis systems. One specific advantage discussed by the authors is that “it more easily allows for rich conditioning on various attributes”. The vowel cluster labels used here are an example of this conditioning.

The “C” in CBHG refers to a bank of 1-dimensional convolutional filters; this layer is what allows Tacotron to easily incorporate the vowel markings as if they were simply another character. The convolutional filters include a width of up to 16 characters (in this case, phonemes plus vowel cluster labels), which is more than enough for the model to learn how successive cluster labels are typically associated with the prosody of the training audio.

This project uses Keith Ito’s open-source implementation of Tacotron, with only small modifications required to accomodate the modified character set. The model was trained with his LJ Speech Dataset [21], which he describes on his website as follows: “This is a public domain speech dataset consisting of 13,100 short audio clips of a single speaker reading passages from 7 non-fiction books. A transcription is provided for each clip. Clips vary in length from 1 to 10 seconds and have a total length of approximately 24 hours. The texts were published between 1884 and 1964, and are in the public domain. The audio was recorded in 2016-17 by the LibriVox project and is also in the public domain.”

The result of this portion of the system is a trained model that takes a labeled phone sequence as input and produces audio as output. For this project, the model was trained for approximately 150,000 steps, which took approximately four days on a Tesla K80 GPU running on Google Cloud.

7. NEW UTTERANCE GENERATION

Once the vowel-cluster and speech synthesis models have been trained, synthesizing new outputs is straightforward. The only difference at this phase is that Kaldi can optionally be used to generate a transcription before computing the alignments. Of course, if the input text is provided along with

the audio, the alignment will be more accurate. Once the alignments are computed, the prosody features for the input utterance are extracted as before and the vowel clustering model is then used to assign cluster labels to each vowel. This sequence of steps results in a labeled phone sequence. The trained Tacotron model then uses this sequence directly to synthesize a new audio file.

8. RESULTS

Although my resources for this project did not permit me to conduct a controlled experiment, a potential one is discussed in Sec 11. However, several pairs of input and output audio are included with the code for this project. These demonstrate, at least to the author’s ears, that the extracted features and resulting cluster labels are indeed capturing some semantic meaning in the input that is reflected in the synthesized audio. They also reveal some shortcomings of the approach taken here and possibilities for future improvement.

The eight cluster centers are displayed in Figure 4.

Here I need to demonstrate that the clusters are in fact “semantically” different in some way. Maybe include some metric of these or run TSNE on the coefficients.

Find some way to quantify that it actually did something beyond “she never stole my money”.

Try and quantify that the different clusters are actually different in some way. This is probably the most important section. Quantify if they are different from male to female speakers in any way.

Try and quantify that the speech result is better for my Tacotron than for without annotations. Say why this could be useful even if no one wants to do speech to speech.

Try and quantify that the output is actually preserving stuff from the original speech dataset.

9. DISCUSSION

9.1. Vowel Clustering

The number of clusters was chosen somewhat arbitrarily; it was intended to be high enough to distinguish the ways that English speakers pronounce any single vowel in isolation, but low enough that the resulting clusters would generalize well across multiple speakers. The use of K-means as opposed to any other unsupervised clustering algorithm was also an arbitrary decision, motivated only by the algorithm’s speed and overall good performance on a wide range of clustering tasks.

By contrast, the choice of an unsupervised algorithm instead of a supervised algorithm was intentional. Although my original intent in this project was to train a classifier that could learn to distinguish between stressed and unstressed syllables, I believe that this would have been unnecessarily limiting.

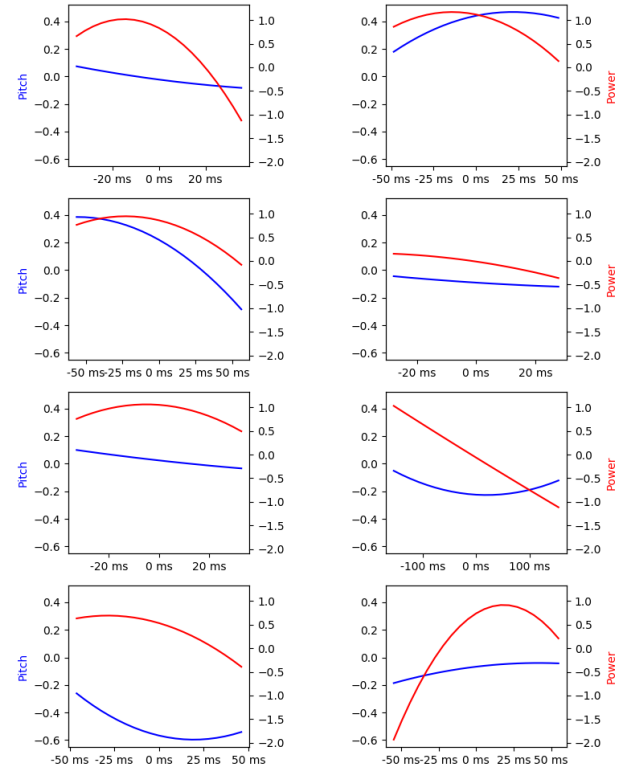


Fig. 4. The eight computed cluster centers. Note that while the scale of the y-axis is constant across plots, the contours have different durations.

Indeed, although Pierrehumbert [17] distinguishes only between H, L, L+H, and H+L for patterns of the F0 contour, the statistics for these patterns may co-occur with the power contour and duration features in meaningful and unknown ways. Rather than limit myself a predefined set of classes, I chose to let the data speak for itself when identifying clusters.

Another choice I made was to compute a single set of clusters for all vowels, as opposed to computing clusters for each vowel individually. Arguments can be made for either choice; to the extent that different vowels have distinct phonetic roles, using a single set of prosody cluster labels for all vowels may inadvertently cause the clusters to distinguish between these vowels instead of between the intended prosodic features. For example, diphthongs may exhibit distinct patterns from monophthongs or the schwa. In my first version of this project, I computed four distinct clusters for each vowel individually. For the final version, I decided to use a common set of clusters based on my belief (as a native English speaker)

that the various pitch and stress patterns are more similar than different from vowel to vowel, and that the system would perform better using more common clusters than fewer specific clusters per vowel.

10. LIMITATIONS

11. FUTURE WORK

I could probably have also just used the actual Legendre coefficients themselves but this would have required tinkering with the Tacotron internals more to accept continuous-valued features as part of the sequence instead of just a one-hot encoded value. This is something that could go in a future work section.

Describe experiment

12. ACKNOWLEDGEMENTS

I would like to thank Keith Ito for his outstanding open-source implementation of Tacotron. This project would not have been possible without his work. I would also like to thank Dan Povey, the lead developer of Kaldi, which was also essential to this project. Finally, I would like to thank Professor Beigi for teaching this class, which has been a pleasure!

13. REFERENCES

- [1] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, “The kaldi speech recognition toolkit,” in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. Dec. 2011, IEEE Signal Processing Society, IEEE Catalog No.: CFP11SRW-USB.
- [2] Elisabeth Selkirk, “Sentence prosody: Intonation, stress, and phrasing,” *The handbook of phonological theory*, vol. 1, pp. 550–569, 1995.
- [3] Pegah Ghahremani, Bagher BabaAli, Daniel Povey, Korbinian Riedhammer, Jan Trmal, and Sanjeev Khudanpur, “A pitch extraction algorithm tuned for automatic speech recognition,” in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2494–2498.
- [4] David Talkin, “A robust algorithm for pitch tracking (rapt),” *Speech coding and synthesis*, vol. 495, pp. 518, 1995.
- [5] Hiroya Fujisaki, “Information, prosody, and modeling-with emphasis on tonal features of speech,” in *Speech Prosody 2004, International Conference*, 2004.
- [6] Siwei Wang and Gina-Anne Levow, “Mandarin chinese tone nucleus detection with landmarks,” in *Ninth Annual Conference of the International Speech Communication Association*, 2008.
- [7] Pui-Fung Wong and Man-Hung Siu, “Decision tree based tone modeling for chinese speech recognition,” in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP’04). IEEE International Conference on*. IEEE, 2004, vol. 1, pp. I-905.
- [8] Chi-Yueh Lin and Hsiao-Chuan Wang, “Language identification using pitch contour information,” in *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP’05). IEEE International Conference on*. IEEE, 2005, vol. 1, pp. I-601.
- [9] Leena Mary, *Extraction and representation of prosody for speaker, speech and language recognition*, Springer Science & Business Media, 2011.
- [10] Rubeena A Khan and JS Chitode, “Concatenative speech synthesis: A review,” *International Journal of Computer Applications*, vol. 136, no. 3, pp. 6, 2016.
- [11] Paul Taylor, *Text-to-speech synthesis*, Cambridge university press, 2009.

- [12] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, “Wavenet: A generative model for raw audio,” *arXiv preprint arXiv:1609.03499*, 2016.
- [13] Sercan O Arik, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Jonathan Raiman, Shubho Sengupta, et al., “Deep voice: Real-time neural text-to-speech,” *arXiv preprint arXiv:1702.07825*, 2017.
- [14] Sercan Arik, Gregory Diamos, Andrew Gibiansky, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou, “Deep voice 2: Multi-speaker neural text-to-speech,” *arXiv preprint arXiv:1705.08947*, 2017.
- [15] Jose Sotelo, Soroush Mehri, Kundan Kumar, Joao Felipe Santos, Kyle Kastner, Aaron Courville, and Yoshua Bengio, “Char2wav: End-to-end speech synthesis,” 2017.
- [16] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al., “Tacotron: A fully end-to-end text-to-speech synthesis model,” *arXiv preprint arXiv:1703.10135*, 2017.
- [17] Janet Breckenridge Pierrehumbert, *The phonology and phonetics of English intonation*, Ph.D. thesis, Massachusetts Institute of Technology, 1980.
- [18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [19] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.
- [20] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.
- [21] Keith Ito, “The lj speech dataset,” <https://keithito.com/LJ-Speech-Dataset/>, 2017.