Using a Recurrent Neural Network and Articulatory Synthesis to Accurately Model Speech Output

Salvatore Skare and Allison Sauppé
Computer Science Department
University of Wisconsin–La Crosse
La Crosse, WI 54601
skare.salvato@uwlax.edu

Abstract

The performance of text-to-speech programs is vital to the adoption of emerging technologies such as virtual assistants and interactive computer systems. However, current systems leave much to be desired. This research aims to construct a text-to-speech system using articulatory synthesis and a recurrent neural network to more accurately model human speech. Some of the aspects of speech of interest to improve upon are prosodic components, enunciation, and pleasantness to listen to. Previously, an articulatory synthesis text-to-speech system has not been feasible due to the complexity of such a system. A human vocal tract simulator must model the frequency and intensity of the glottal sound wave, as well as the perturbations caused by the vocal tract and nasal pathway. It must also model how the vocal tract changes in diameter, area and length at multiple locations. A neural network is well suited to providing these many inputs to a speech synthesizer at a rapid rate. A recurrent neural network is also capable of responding to context dependent prosodic and linguistic issues. In this research two different methods of articulatory synthesis were tested with a recurrent neural network to generate the inputs. The results show that these types of networks are capable of learning how to generate the parameters for speech.

1 Introduction

Text to speech (TTS) programs are a class of computer program that takes typed text and outputs it as spoken words. Although programs to turn text into speech output have existed for decades, with the rising popularity of smartphone assistants and other devices interfaced with solely via voice, TTS programs are seeing more use than ever. However, even the best TTS programs produce robotic and choppy sounding speech that still sounds machine-like. This can affect the way a user interacts with and perceives these programs in a negative way.

These issues create a barrier to holding long conversations with virtual assistants and successfully conveying long sentences and paragraphs with TTS. Creating a TTS system that can accurately transcribe text as well as being pleasant to listen to would aid in making spoken communication from computers more user-friendly. This would have applications such as improved in-car information and entertainment systems, personal digital assistants, smart devices/Internet of Things, and accessibility services for the blind.

The majority of TTS programs today use concatenative models, translating the text into individual phonemes and trying to concatenate them in a pleasing way, which can be difficult for a computer to do. For example, context dependent pronunciations of words like "project" provide issues for TTS systems (e.g. "working on a project" versus "projecting one's voice"). In recent years, this issue has been tackled using probabilistic heuristic methods such as Hidden Markov Models, which look at previous and next words to predict the correct pronunciation. Even more recently is the invention of Google's WaveNet, which uses a convolutional neural network to generate audio from phonemes at the sound wave level [1].

The largest remaining barrier to accurate TTS programs is prosody, which is defined as "the melody, rhythm, and emphasis of the speech at the perceptual level" [2]. In a paper from Berkeley Speech Technologies, O'Malley explains how with traditional TTS the lack of prosody can cause the speech to sound boring, but attempts to add more variation result in a "foolish-sounding" voice [3].

An alternative to concatenative speech synthesis, articulatory synthesis, provides a way to mitigate some of these issues. Lemmetty described articulately synthesis as follows:

> Articulatory synthesis tries to model the human vocal organs as perfectly as possible, so it is potentially the most satisfying method to produce high-quality synthetic speech. On the other hand, it is also one of the most difficult methods to implement and the computational load is also considerably higher than with other common methods [2].

Because of the numerous parameters that need to be updated in real-time to produce high-quality articulatory speech synthesis, an algorithm to do so has not yet been invented.

2 Methods

In this research two methods of producing speech from models of the human vocal system were tested with a machine learning approach, 1) a vocal synthesizer based on mathematical models of the vocal system and air flow, and 2) a formant-based synthesizer that instead uses a series of formant functions applied to an excitation source to produce vocalizations. Both methods were trained on single phoneme recordings and the LibriSpeech corpus of speech data. All software written for this research and described as follows is available online[1].

2.1 Vocal Synthesizer

To create the vocal synthesizer, a combination of the LF-model for glottal waves and the vocal tract model from Story [4] was used. The LF-model simulates the pressure waves at the human glottis from a set of timing parameters [5]. The glottal wave is created by a piecewise function, defined as follows:

$$g(t) = E_0 e^{\alpha t} \sin(\omega_g t), 0 \leq t \leq T_e \quad \text{(Initial phase)}$$

$$g(t) = -\frac{E_e}{\epsilon T_a}[e^{-\epsilon(t-T_e)} - e^{-\epsilon(T_c-T_e)}], T_e < t < T_c \leq T_0 \quad \text{(Return phase)}$$

The direct synthesis parameters, (E_e, E_0, α, ω, ϵ), are derived from the timing parameters, listed in Table 1.

Parameter	Definition
T_0	Fundamental period
T_c	Ending of the return phase
T_a	Duration of the return phase
T_e	Instant of glottal closure

Table 1: Timing parameters from the LF-model used to generate glottal wave

Using the linear regression from the transformed LF-model described by Fant, Liljencrants, and Lin [6] we are able to predict timing parameters for the LF-model from just a wave-shape parameter, R_d, and the period, T_0. From these two variables we are able to create the glottal wave portion of speech. An example of a glottal wave generated by the synthesizer is shown in Figure 1.

[1] `https://gitlab.com/saljs/rnn-articulatory-tts`

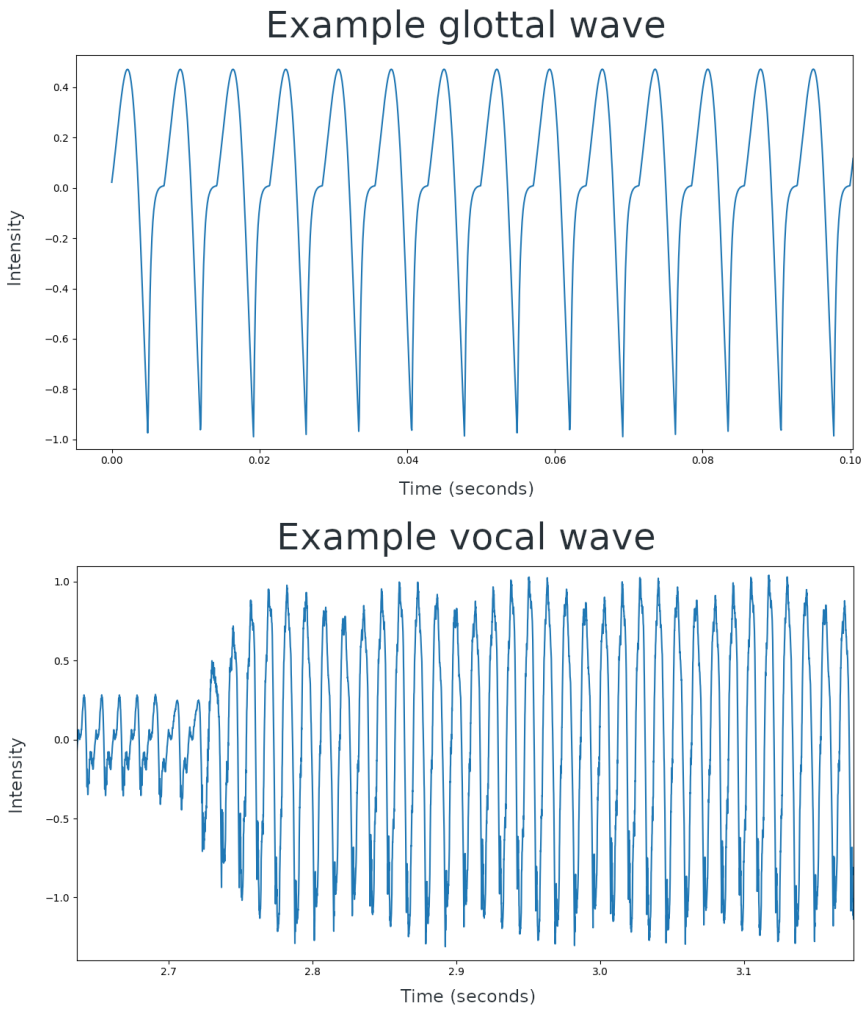

Figure 1: Top: An example glottal wave produced by the vocal synthesizer. Bottom: An example of the glottal wave above after being perturbed by the simulated vocal tract.

Figure 2: A 44 section vocal tract representation using the resting diameters given in Appendix A of Story, 2005 [4]. Each section acts as a resonator perturbing the glottal wave.

In order to produce vocalizations, the glottal wave is perturbed by the vocal tract. This is achieved using the parametric model described in Story, 2005. This model achieves perturbation through a series of tiers: area, length, and nasalization [4]. Each tier describes

changes to a segmented vocal tract (Figure 2) based on control parameters, such as length change at glottis and lips, position and magnitude of constrictions, and area of nasal coupling. A full list of inputs to the synthesizer is listed in Table 2. The glottal wave is then sampled through these segments, treating them as a series of resonators. An example of the resulting wave is pictured in Figure 1.

Input	Definition
t	Time to hold the sound for (seconds)
F_0	Frequency (Hz)
R_d	Glottal wave-shape
q_1,q_2	Amplitude coefficients
l_c	Location of consonant constriction
a_c	Area of consonant constriction (cm^2)
r_c	Range of consonant constriction
s_c	Skewing quotient of consonant contraction
m_c	Magnitude of consonant contraction
p_g	Length change at glottal end of vocal tract (cm)
l_g	Center of the length change p_g
r_g	Range of the length change p_g
p_m	Length change at lips (cm)
l_m	Center of the length change p_m
r_m	Range of the length change p_m
a_{np}	Cross-sectional area of nasal coupling port (cm^2)

Table 2: The list of parameters the synthesizer takes as inputs.

The combined glottal and vocal synthesizer take a set of 17 inputs (Table 2). These inputs are generated by a recurrent neural network (RNN). The structure of this network is described in Figure 3. Each input sequence of text to the RNN was converted into phonemes, encoded into a one-hot vector, and broken into discrete time-steps of one phoneme each. At the final layer of the RNN, 17 outputs are generated then become the inputs to the vocal synthesizer. The synthesizer then outputs a sound wave sampled at 16kHz of between 0 and 60 seconds in length. During the training process, this sound wave is normalized, then compared to the training sample using a mean squared error metric. This error term is then backpropagated through the network.

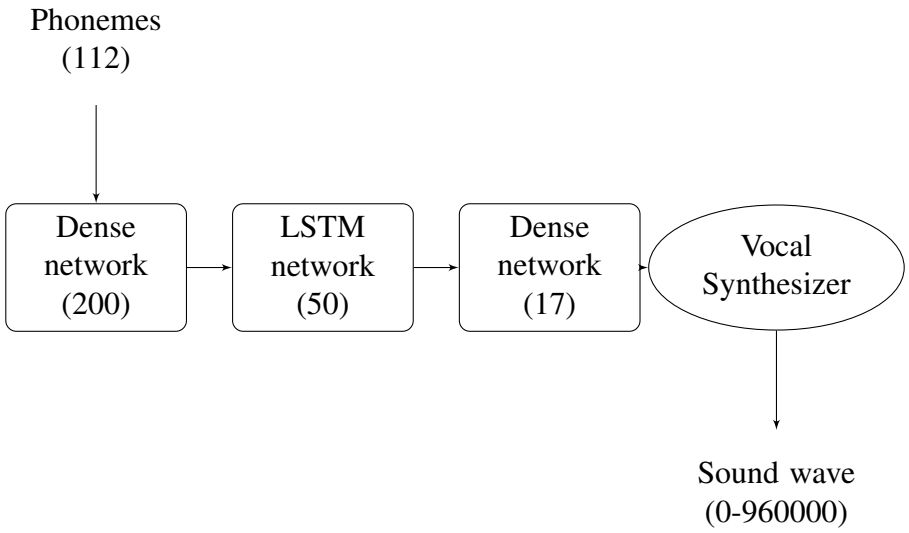

Figure 3: The recurrent neural network used for the vocal synthesizer. Blocks represent multiple layers and ovals represent single layers/operations. The output size at each level is listed in parenthesis.

2.2 Formant Synthesizer

The formant synthesizer, like the vocal synthesizer, has two main components: a source excitation and a series of perturbations caused by the vocal tract. Unlike the vocal synthesizer, instead of applying perturbations by keeping track of the state of a vocal tract, a series of formant functions are applied. The benefit of this synthesis method is that it is computationally much faster then the previous synthesizer. Using linear predictive coding, the excitation and formant functions can be extracted from a recorded speech sample. The Speech Signal Processing Toolkit software package was used in this research for this process [7]. This means that instead of creating an end-to-end trainable model, a RNN can be trained to generate these two sequences without having to generate a final sound wave in order to calculate an error value for backpropagation. This drastically reduces the time needed to train the model.

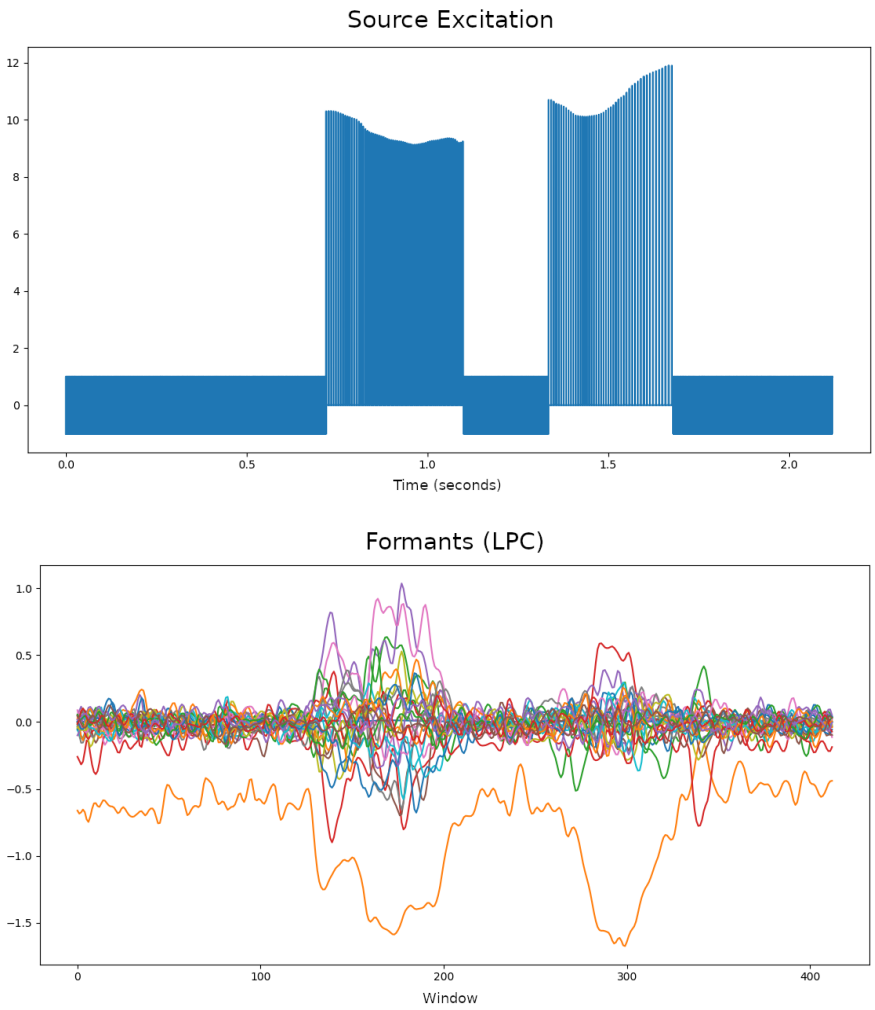

Figure 4: An example of the utterance “Hello world” split into excitation (top) and formant functions (bottom).

Inputs are provided to the formant synthesizer RNN in the same way as with the vocal synthesizer. The input layer then feeds to two separate recurrent networks made up of LSTM cells. The output from these layers are then fed to deconvolutional layers [8], which serve to identify and extrapolate patterns into output sequences. The excitation output is one dimensional, but the number of formant functions depends on the selected order (in this research an order of 25 was used), so the deconvolutional layers that create the formant functions are two dimensional, with a number of filters equal to the order plus one for the fundamental formant frequency. The complete network is described in Figure 5.

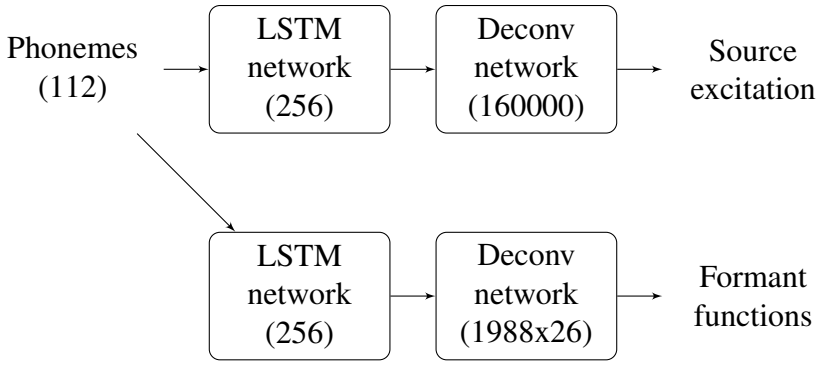

Figure 5: The recurrent neural network used for the formant synthesizer. Blocks represent multiple layers. The output size at each level is listed in parenthesis.

3 Results

When trained on the single phoneme data set, both synthesizers were able to achieve accuracy's of $> 95\%$ compared to source recordings. The formant synthesizer exhibits a better learning curve, while the vocal synthesizer takes longer to reach 95% accuracy and oscillates more (Figure 6).

The formant synthesizer outperforms the vocal synthesizer when trained on the LibriSpeech corpus. The vocal synthesizer was unable to converge to a high level of accuracy, averaging only about 70%, while the format synthesizer converged to $> 99\%$ accuracy.

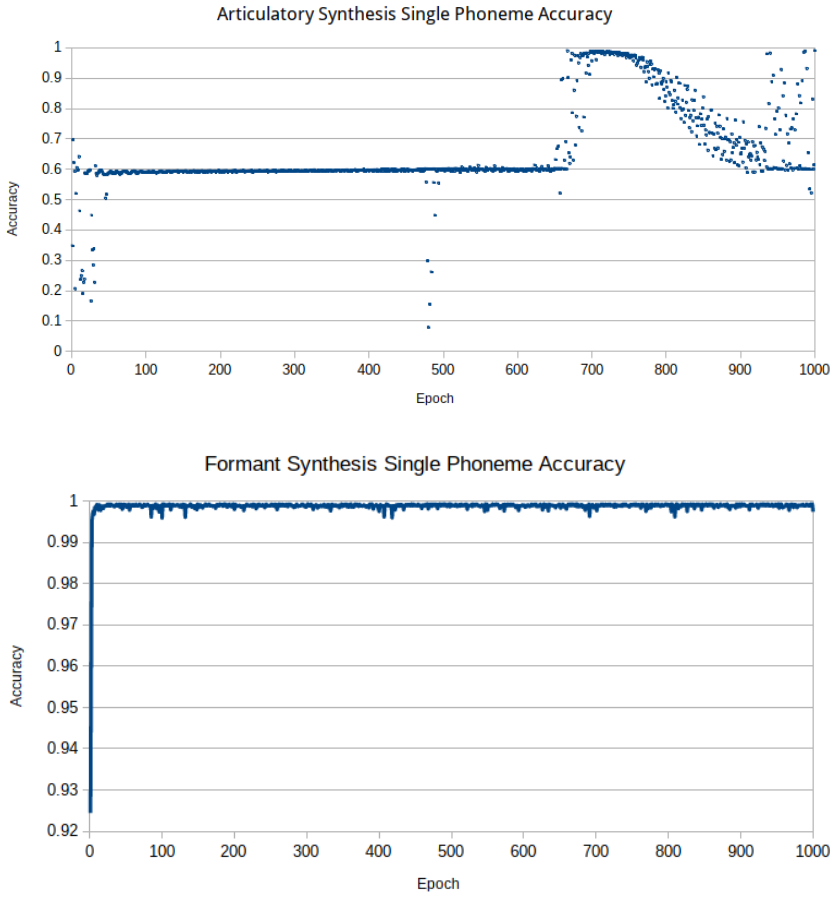

Figure 6: Accuracy of both synthesizers when trained on single phoneme data over time. The data points are connected on the formant synthesizer plot to better show the trend over time.

The reason the formant synthesizer outperforms the vocal synthesizer likely has to do with how error is backpropagated though the network. To speed up the synthesis of sound waves with the vocal synthesizer, a buffer of 256 samples is returned for every set of inputs. This makes estimating partial derivatives for each input parameter impossible, so the error is backpropagated inaccurately. Another version of the synthesizer that produced one sample at a time was developed and tested, but ended up being prohibitively slow. The formant synthesizer, using a non-end-to-end trainable network, didn't suffer from this issue and was able to learn to produce speech parameters more effectively.

4 Discussion

One of the main limitations of this research was that only a limited number of RNN configurations were evaluated. Future research could attempt to improve these results with other network configurations. Additionally, using generative neural networks, such as a generative adversarial network would be another avenue to pursue further [9].

In a full TTS system using the formant synthesizer described, the generated excitation and formant functions would need to be synthesized into a vocal wave. The number of formant functions used in this research is the minimum needed to produce high quality speech. A full TTS system might also increase this number.

The results of this research demonstrate that recurrent neural networks are capable of generating parameters to be used in articulatory speech synthesis. A high-quality articulatory synthesis TTS system would have wide ranging benefits in numerous applications and would serve to create more user-friendly voice interfaces.

5 Acknowledgments

I would like to acknowledge my faculty mentor, Dr. Allison Sauppé, who has helped me at every step of this research. In addition I would like to thank Dr. Martin Allen who provided valuable insight and instruction into the inner workings and applications of neural networks. This research would not have been possible without generous funding from the Dean's Distinguished Fellow's grant from the University of Wisconsin La Crosse College of Science and Health. Lastly, I would like to thank the University of Wisconsin–La Crosse Computer Science Department for providing the equipment necessary to perform this research.

References

[1] Aaron van den Oord, Sander Dieleman, Sander Dieleman, et al. *WaveNet: A Generative Model for Raw Audio*. Tech. rep. Google DeepMind, London, UK, 2016.

[2] Sami Lemmetty. "Review of Speech Synthesis Technology". MA thesis. Helinski University of Technology, 1999. URL: http://research.spa.aalto.fi/publications/theses/lemmetty_mst/.

[3] M. H. O'Malley. "Text-to-speech conversion technology". In: *Computer* 23.8 (1990), pp. 17–23. ISSN: 0018-9162. DOI: 10.1109/2.56867.

[4] Brad H. Story. "A parametric model of the vocal tract area function for vowel and consonant simulation". In: *The Journal of the Acoustical Society of America* 117 (Jan. 2005), pp. 3231–3254. DOI: 10.1121/1.1869752.

[5] G Fant, J Liljencrants, and Qiguang Lin. "A Four-Parameter Model of Glottal Flow". In: *STL-QPSR* 4 (Jan. 1985).

[6] G Fant, J Liljencrants, and Qiguang Lin. "The LF-model revisited. Transformations and frequency domain analysis". In: *KTH, Speech Transmission Laboratory, Quarterly Report* 2-3 (Jan. 1995), pp. 119–156.

[7] Satoshi Imai, Takao Kobayashi, and Keiichi Tokuda. *Speech Signal Processing Toolkit (SPTK)*. http://sp-tk.sourceforge.net/.

[8] M. D. Zeiler, D. Krishnan, G. W. Taylor, et al. "Deconvolutional networks". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. June 2010, pp. 2528–2535. DOI: 10.1109/CVPR.2010.5539957.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.