CHAR2WAV: END-TO-END SPEECH SYNTHESIS

Jose Sotelo, Soroush Mehri, Kundan Kumar*, Joao Felipe Santos†, Kyle Kastner, Aaron Courville‡, Yoshua Bengio§
Université de Montréal
* IIT Kanpur
† INRS-EMT
‡ CIFAR Fellow
§ Senior CIFAR Fellow

ABSTRACT

We present Char2Wav, an end-to-end model for speech synthesis. Char2Wav has two components: a reader and a neural vocoder. The reader is an encoder-decoder model with attention. The encoder is a bidirectional recurrent neural network that accepts text or phonemes as inputs, while the decoder is a recurrent neural network (RNN) with attention that produces vocoder acoustic features. Neural vocoder refers to a conditional extension of SampleRNN which generates raw waveform samples from intermediate representations. Unlike traditional models for speech synthesis, Char2Wav learns to produce audio directly from text.

1 INTRODUCTION

The main task in speech synthesis consists of mapping text to audio signal. There are two primary goals in speech synthesis: intelligibility and naturalness. Intelligibility describes the clarity of the synthesized audio, specifically how well a listener is able to extract the original message. Naturalness describes information not directly captured by intelligibility, such as overall ease of listening, global stylistic consistency, regional or language level nuances, among others.

With traditional speech synthesis approaches, this task has been accomplished by dividing the problem into two stages. The first stage, known as the frontend, transforms the text into linguistic features. These linguistic features usually include phone, syllable, word, phrase and utterance-level features (Zen, 2006; Zen et al., 2013; van den Oord et al., 2016). The second stage, known as the backend, takes as input the linguistic features generated by the frontend and produces the corresponding sound. WaveNets (van den Oord et al., 2016) are a high quality approach to a "neural backend".

For a more detailed review of traditional models for speech synthesis, we recommend consulting Taylor (2009). In this work, we integrate the frontend and the backend and learn the whole process end-to-end. This procedure eliminates the need for expert linguistic knowledge, which removes a major bottleneck in creating synthesizers for new languages. Defining good linguistic features is often time-consuming and language specific. We use a powerful model to learn this information from the data.

2 RELATED WORK

Attention based models have been previously used in machine translation (Cho et al., 2014; Bahdanau et al., 2015), speech recognition (Chorowski et al., 2015; Chan et al., 2016), and computer vision (Xu et al., 2015) among other applications. Our work has been heavily influenced by the work of Alex Graves (Graves, 2013; Graves, 2015). In a guest lecture Graves demonstrated a speech synthesis model using an attention mechanism, an extension of his previous work on handwriting generation. Unfortunately, the speech extension was never published, so we cannot directly compare our approach to his work. However, his results were a key inspiration to us, and we hope that this work can be useful as a starting point for further developments in end-to-end speech synthesis.
3 Model Description

3.1 Reader

We adopt the notation of Chorowski et al. (2015). An attention-based recurrent sequence generator (ARSG) is a recurrent neural network that generates a sequence \( Y = (y_1, \ldots, y_T) \) conditioned on an input sequence \( X \). \( X \) is preprocessed by an encoder that outputs a sequence \( h = (h_1, \ldots, h_L) \). In this work, the output \( Y \) is a sequence of acoustic features and \( X \) is the text or the phoneme sequence to be generated. Furthermore, the encoder is a bidirectional recurrent network.

At the \( i \)-th step the ARSG focuses on \( h \) and generates \( y_i \):

\[
\alpha_i = \text{Attend}(s_{i-1}, \alpha_{i-1}, h) \\
g_i = \sum_{j=1}^{L} \alpha_{i,j} h_j \\
y_i \sim \text{Generate}(s_{i-1}, g_i) \\
s_i = RNN(s_{i-1}, g_i, y_i)
\]

where \( s_{i-1} \) is the \((i-1)\)-th state of the generator recurrent neural network and \( \alpha_i \in \mathbb{R}^L \) are the attention weights or alignment.

In this work, we use the location-based attention mechanism developed by Graves (2013). We have \( \alpha_i = \text{Attend}(s_{i-1}, \alpha_{i-1}) \) and given a length \( L \)-conditioning sequence \( h \), we have:

\[
\phi(i, l) = \sum_{k=1}^{K} \rho_i^k \exp(-\beta_i^k (\kappa_i^k - l)^2) \\
\alpha_i = \sum_{l=1}^{L} \phi(i, l)
\]

where \( \kappa_i \), \( \beta_i \), and \( \rho_i \) represent the location, width and importance of the window respectively.

3.2 Neural Vocoder

Speech generation using a vocoder is limited by the reconstruction quality of that specific vocoder. Because of that, we replace the vocoder with a learned parametric neural module. We use SampleRNN (Mehri et al., 2016) as an enhanced function approximator for this purpose.

SampleRNN has recently been proposed to model extremely long-term dependencies in sequential data such as audio signals. The hierarchical structure in SampleRNN is designed to capture dynamics of a sequence at different time scales. This is necessary to capture long range correlations between distant audio timesteps (e.g. word-level correlations in speech signals) as well as nearby audio timesteps dynamics.

We use a conditional version of the same model to learn the mapping from a sequence of vocoder features to corresponding audio samples. Each vocoder feature frame is added as an extra bias to the corresponding state in the top tier. This allows the module to use the past audio samples and vocoder feature frames to generate the current audio samples.
4 RESULTS

We do not provide a comprehensive quantitative analysis of results at this time. Instead, we provide samples from our model[1]. In Figure 2, we show samples generated by our model and their corresponding alignments to the text.

Figure 2: Samples from the models conditioned on a) English phonemes, b) English text and c) Spanish text. The models for a) and b) were trained on the VCTK dataset (Yamagishi, 2012) whereas the model for c) was trained on the DIMEX-100 dataset (Pineda et al., 2010).

ACKNOWLEDGMENTS

We thank the developers of Merlin (Wu et al., 2016), Theano (Theano Development Team, 2016), and Blocks (van Merriënboer et al., 2015). This work used the computational resources provided by Compute Canada and Calcul Québec. João Felipe Santos and Jose Sotelo thank the Fonds de Recherche du Québec - Nature et Technologies (FQRNT) for their support. Jose Sotelo also thanks the Consejo Nacional de Ciencia y Tecnología (CONACyT) as well as the Secretaría de Educación Pública (SEP) for their support. We thank the patient explanations of Dzmitry Bahdanau. His comments greatly improved this work. We also thank Alexandre de Brébisson, Jonathan Lucuix-André, Laurent Dinh, Ishaan Gulrajani, Ritesh Kumar, David Warde-Farley, Eugene Belilovsky, Natasha Jacques, Tim Cooijmans, Anna Huang, and Junyoung Chung for their comments and suggestions.

REFERENCES


