A Real-Time Wideband Neural Vocoder at 1.6 kb/s Using LPCNet

Jean-Marc Valin[1]*, Jan Skoglund*[2]

[1]Mozilla, Mountain View, CA, USA
[2]Google LLC, San Francisco, CA, USA

jmvalin@jmvalin.ca, jks@google.com

Abstract

Neural speech synthesis algorithms are a promising new approach for coding speech at very low bitrate. They have so far demonstrated quality that far exceeds traditional vocoders, at the cost of very high complexity. In this work, we present a low-bitrate neural vocoder based on the LPCNet model. The use of linear prediction and sparse recurrent networks makes it possible to achieve real-time operation on general-purpose hardware. We demonstrate that LPCNet operating at 1.6 kb/s achieves significantly higher quality than MELP and that uncompressed LPCNet can exceed the quality of a waveform codec operating at low bitrate. This opens the way for new codec designs based on neural synthesis models.

Index Terms: neural speech synthesis, wideband coding, vocoder, LPCNet, WaveRNN

1. Introduction

Very low bitrate parametric codecs have existed for a long time [1, 2], but their quality has always been severely limited. While they are efficient at modeling the spectral envelope (vocal tract response) of the speech using linear prediction, no such simple model exists for the excitation. Despite some advances [3, 4, 5], modeling the excitation signal has remained a challenge. For that reason, parametric codecs are rarely used above 3 kb/s.

Neural speech synthesis algorithms such as Wavenet [6] and SampleRNN [7] have recently made it possible to synthesize high quality speech. They have also been used in [8, 9] (WaveNet) and [10] (SampleRNN) to synthesize high-quality speech from coded features, with a complexity in the order of 100 GFLOPS. This typically makes it impossible to use those algorithms in real time without high-end hardware (if at all). In this work, we focus on simpler models, that can be implemented on general-purpose hardware and mobile devices for real-time communication, and that work for any speaker, in any language. Moreover, we target the very low bitrate of 1.6 kb/s, which is beyond the reach of conventional waveform speech coders.

To reduce computational complexity, WaveRNN [11] uses a sparse recurrent neural network (RNN). Other models use linear prediction to remove the burden of spectral envelope modeling from the neural synthesis network [12, 13, 14]. That includes our previous LPCNet work [12] (summarized in Section 2), which augments WaveRNN with linear prediction to achieve low complexity speaker-independent speech synthesis.

We now address quantization of the LPCNet features to achieve low-bitrate speech coding (Section 3). Section 4 discusses training considerations, and Section 5 presents our results. We conclude with ideas for improvement in Section 6.

2. LPCNet Overview

The WaveRNN model [11] is based on a sparse gated recurrent unit (GRU) [15] layer. At time t, it uses the previous audio sample s_{t-1}, as well as frame conditioning parameters to generates a discrete probability distribution $P\left(s_t\right)$ from which the output s_t is sampled. LPCNet [12] improves on WaveRNN by adding linear prediction, as shown in Fig. 1. It is divided in two parts: a frame rate network that computes conditioning features for each 10-ms frame, and a sample rate network that computes conditional sample probabilities. In addition to using the previously generated speech sample s_{t-1}, LPCNet also uses the 16^{th} order prediction $p_t = \sum_{i=1}^{16} a_i s_{t-i}$ and the previously generated excitation e_{t-1}, where $e_t = s_t - p_t$.

LPCNet operates on signals quantized using 256-level μ-law. To avoid audible quantization noise we apply a pre-emphasis filter $E(z) = 1 - \alpha z^{-1}$ on the input speech (with $\alpha = 0.85$) and the inverse (de-emphasis) filter on the output. This shapes the noise and makes it less perceptible. Considering that s_{t-1}, p_t, and e_{t-1} are discrete, we can pre-compute the contribution $\mathbf{v}_i^{(\cdot,\cdot)}$ of each possible value to the $\mathrm{GRU_A}$ gates and state so that only lookups are necessary at run-time. In addition, the contribution $\mathbf{g}^{(\cdot)}$ of the frame rate network to $\mathrm{GRU_A}$ can be computed only once per frame. After these simplifications, only the recurrent matrices $\mathbf{W}_{(\cdot)}$ remain and the sample rate network is computed as

$$\begin{aligned}
\mathbf{u}_t =& \sigma\left(\mathbf{W}_u \mathbf{h}_{t-1} + \mathbf{v}_{s_{t-1}}^{(u,s)} + \mathbf{v}_{p_t}^{(u,p)} + \mathbf{v}_{e_{t-1}}^{(u,e)} + \mathbf{g}^{(u)}\right) \\
\mathbf{r}_t =& \sigma\left(\mathbf{W}_r \mathbf{h}_{t-1} + \mathbf{v}_{s_{t-1}}^{(r,s)} + \mathbf{v}_{p_t}^{(r,p)} + \mathbf{v}_{e_{t-1}}^{(r,e)} + \mathbf{g}^{(r)}\right) \\
\widetilde{\mathbf{h}}_t =& \tau\left(\mathbf{r}_t \circ \left(\mathbf{W}_h \mathbf{h}_{t-1}\right) + \mathbf{v}_{s_{t-1}}^{(h,s)} + \mathbf{v}_{p_t}^{(h,p)} + \mathbf{v}_{e_{t-1}}^{(h,e)} + \mathbf{g}^{(h)}\right) \\
\mathbf{h}_t =& \mathbf{u}_t \circ \mathbf{h}_{t-1} + \left(1 - \mathbf{u}_t\right) \circ \widetilde{\mathbf{h}}_t \\
P\left(e_t\right) =& \operatorname{softmax}\left(\text{dual_fc}\left(\mathrm{GRU_B}\left(\mathbf{h}_t\right)\right)\right),
\end{aligned} \tag{1}$$

where $\sigma\left(x\right)$ is the sigmoid function, $\tau\left(x\right)$ is the hyperbolic tangent, $\circ$ denotes an element-wise vector multiply, and $\mathrm{GRU_B}\left(\cdot\right)$ is a regular (non-sparse) GRU. The dual fully-connected (dual_fc) layer is defined as

$$\text{dual\_fc}(\mathbf{x}) = \mathbf{a}_1 \circ \tau\left(\mathbf{W}_1 \mathbf{x}\right) + \mathbf{a}_2 \circ \tau\left(\mathbf{W}_2 \mathbf{x}\right), \tag{2}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are weight matrices and $\mathbf{a}_1$ and $\mathbf{a}_2$ are scaling vectors.

Throughout this paper, biases are omitted for clarity. The synthesized excitation sample e_t is obtained by sampling from the probability distribution $P\left(e_t\right)$ after lowering the *temperature* of voiced frames as described in eq. (7) of [12]. As a way of reducing the complexity, $\mathrm{GRU_A}$ uses sparse recurrent matrices with non-zero blocks of size 16x1 to ensure efficient vectorization. Because the hidden state update is more important than the reset and update gates, we keep 20% of the weights in $\mathbf{W}_h$, but only 5% of those in $\mathbf{W}_r$ and $\mathbf{W}_u$, for an average of 10%.