Improving Opus Low Bit Rate Quality with Neural Speech Synthesis

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

[1]Google, San Francisco, CA, USA
[2]Amazon, Palo Alto, CA, USA

jks@google.com, jmvalin@amazon.com

Abstract

The voice mode of the Opus audio coder can compress wideband speech at bit rates ranging from 6 kb/s to 40 kb/s. However, Opus is at its core a waveform matching coder, and as the rate drops below 10 kb/s, quality degrades quickly. As the rate reduces even further, parametric coders tend to perform better than waveform coders. In this paper we propose a backward-compatible way of improving low bit rate Opus quality by re-synthesizing speech from the decoded parameters. We compare two different neural generative models, WaveNet and LPCNet. WaveNet is a powerful, high-complexity, and high-latency architecture that is not feasible for a practical system, yet provides a best known achievable quality with generative models. LPCNet is a low-complexity, low-latency RNN-based generative model, and practically implementable on mobile phones. We apply these systems with parameters from Opus coded at 6 kb/s as conditioning features for the generative models. A listening test shows that for the same 6 kb/s Opus bit stream, synthesized speech using LPCNet clearly outperforms the output of the standard Opus decoder. This opens up ways to improve the decoding quality of existing speech and audio waveform coders without breaking compatibility.

Index Terms: WaveNet, LPCNet, Opus, neural vocoder

1. Introduction

Speech compression methods operating at very low bit rates often represent speech as a sequence of parameters extracted at the encoder. These systems are referred to as parametric coders or vocoders. Speech generated at the decoder from the transmitted parameters sounds *similar* to the original speech, but the waveform does not match the original. The resulting speech often sounds intelligible but with a robotic character. Examples are linear predictive vocoders [1, 2] or sinusoidal coders [3, 4]. Another family of coders is the hybrid waveform coders, which use some signal modelling, yet try to mimic the signal waveform. A typical example is code excited linear prediction (CELP) [5]. Hybrid coders are used in most mobile telephony and VoIP standards, examples are the AMR coder [6, 7] and the IETF Internet coder Opus [8]. However, because these schemes attempt to reconstruct the signal waveform, they require higher rates to be successful, and at very low rates, say below 6 kb/s, the quality of waveform matching hybrid coders eventually becomes inferior even to the quality of parametric coders.

Generative systems using neural speech synthesis have recently demonstrated the ability to produce high quality speech. The first method shown to provide excellent speech quality was WaveNet [9], originally proposed for text-to-speech synthesis. Since then, WaveNet has been used for parametric coding that significantly out-perform more traditional vocoders, either using an existing vocoder bit stream [10], or with a quantized

learned representation set [11]. A typical WaveNet configuration requires a very high algorithmic complexity, in the order of hundreds of GFLOPS, along with a high memory usage to hold the millions of model parameters. Combined with the high latency, in the hundreds of milliseconds, this renders WaveNet impractical for a real-time implementation. Replacing the dilated convolutional networks with recurrent networks improved memory efficiency in SampleRNN [12], which was shown to be useful for speech coding in [13]. WaveRNN [14] also demonstrated possibilities for synthesizing at lower complexities compared to WaveNet. Even lower complexity and real-time operation was recently reported using LPCNet [15].

These previously proposed systems are all based on quantized parametric speech coder features as conditioning to the neural speech generation. In this work, we demonstrate the ability of generative networks to improve the synthesis quality of the hybrid, waveform-matching, Opus coder (Section 2) operating at a low bit rate. The goal here is to improve the quality of an existing waveform coder without changing the bit stream. The results may hopefully encourage the use of neural synthesis to improve the quality of other standard coders at low rates, since deploying new coders to replace existing ones is a long and complicated process. This approach can thus help extending the life of existing coders without introducing compatibility issues during transitions.

For this task, we consider both WaveNet and LPCNet models in Section 3. Section 4 describes the conditioning features and training procedure, and we evaluate the two models in Section 5. We then conclude in Section 6.

2. Opus speech compression

Opus [8] is a versatile coder that supports narrowband (8 kHz sampling frequency) to fullband (48 kHz sampling frequency) speech and audio. For speech communication it has hundreds of millions of users through applications such as Zoom and WebRTC [16, 17] based ones, such as Microsoft Teams, Google Meet, Duo, Skype, and Slack. Opus is also one of the main audio coders used in YouTube for streaming.

It is based on a combination of a linear predictive coding part (SILK [18]) and a transform coding part (CELT [19]). In this work we focus on wideband speech, i.e., a sampling frequency of 16 kHz, using the SILK-only mode of Opus.

For wideband speech, SILK uses 16th-order linear prediction coefficients (LPC). The long-term predictor (LTP) uses a 5-tap filter, which both controls the amount of prediction as a function of frequency and, to some extent, provides some of the benefits of a fractional pitch period. The Opus reference encoder[1] we use for this work jointly optimizes the LPC and LTP to minimize the residual signal variance.

---

* This work was performed while the author was with Mozilla.

[1]https://opus-codec.org/

SILK is a linear prediction-based coder that uses noise feedback coding (NFC) [20] rather than regular CELP [5]. The residual is coded as a sum of pulses, plus a pulses-dependent dither signal. Note that even though there is technically a separation into a spectral envelope filter and an excitation, both SILK and CELP coders are indeed hybrid waveform coders, with weighted waveform matching loss functions.

Because SILK uses entropy coding, it is fundamentally also a variable bit rate (VBR) coder. Rather than having a fixed bit allocation for its quantizers, it uses rate-distortion optimization (RDO) for both the filter and the residual. The number of bits allocated to the filter (represented as line spectral pairs [21]) does not vary significantly with the total bit rate. As a result, the number of bits used for the residual goes down rapidly as the total bit rate decreases below 8 kb/s, with the quality eventually becoming unacceptable. By using a neural synthesis on the Opus bit stream, it is possible to create a decoder that degrades more gracefully without breaking compatibility.

3. Autoregressive Generative Networks for Speech Synthesis

Autoregressive neural synthesis systems are based on the idea that the speech signal probability distribution $p(\mathbf{S})$ can be factorized as a product of conditional probabilities [9]

$$p(\mathbf{S}) = \prod_{t=1}^{T} p(s_t | s_{t-1}, s_{t-2}, \ldots, s_2, s_1), \qquad (1)$$

where $\mathbf{S} = \{s_1, s_2, \ldots, s_T\}$ is a set of consecutive speech samples. The probability of each speech sample s_t is then conditioned on previous samples, i.e., a tractable *scalar* autoregressive structure $p(s_t | s_{t-1}, s_{t-2}, \ldots)$. A practical speech generation system will also need additional conditioning features, $\boldsymbol{\theta}_t$, to guide the waveform generation. Examples of such features are spectral envelope information, pitch, and gain. The output sample s_t is then drawn from the distribution $p(s_t | s_{t-1}, s_{t-2}, \ldots, \boldsymbol{\theta}_t)$, modelled through a neural net as $p(s_t | u(\boldsymbol{\theta}_t, s_{t-1}, s_{t-2}, \ldots, \boldsymbol{\omega}))$, where u denotes a deterministic neural network with parameters (e.g., weights) $\boldsymbol{\omega}$. Examples of distributions utilized in generative systems are discrete softmax [22] and mixtures of logistics [23].

In this work, we explore two autoregressive models for synthesizing speech from Opus parameters. We use WaveNet as an "informal upper bound" that demonstrates the highest obtainable quality from generative networks. To demonstrate what can currently be achieved in real time on general purpose hardware, we also explore LPCNet as generative synthesis.

3.1. WaveNet

The WaveNet architecture is a deep multi-layer structure using dilated convolution with gated cells. The number of layers are typically more than 25 and the conditional variables are supplied to all layers of the network.

A convolutional neural network has a finite memory - the receptive field - which depends on the number of layers in the network. During training, WaveNet learns the parameters of a (discretized) mixture of logistics function that represents the conditional discrete probability distribution $p(s_t)$. The WaveNet architecture has shown impressive speech quality for text-to-speech [9] and low bit rate speech coding [10, 11]. This performance comes at the price of a high complexity, typically 100+ GFLOPS, high memory requirements with millions of

network parameters, and a high latency, 400+ ms. Even though more recent generative architectures, such as WaveRNN [14], have shown the ability to operate at a low complexity, it comes at the cost of trading off quality. At this low complexity the recurrent architectures have not been able to fully match the quality of the original WaveNet. We use a WaveNet model with 27 layers (9 dilation steps) and 256 hidden states in each layer, and with a receptive field of 192 ms. The output of the network is a logistic mixture distribution sampled to produce wideband speech at a 16-bit resolution.

3.2. LPCNet

The WaveRNN model [14] is based on a sparse gated recurrent unit (GRU) [24] layer. LPCNet [15] improves on WaveRNN by adding linear prediction, as shown in Fig. 1. Linear prediction is long known [25] to represent the spectral envelope of speech very well, and this enables the non-linear components of the network to focus on a spectrally flat excitation waveform. LPCNet is divided in two parts: a *frame rate network* that computes conditioning features for each frame, and a *sample rate network* that computes conditional sample probabilities. In addition to using the previously generated speech sample s_{t-1}, the sample rate network also uses the 16th order prediction $y_t = \sum_{i=1}^{16} a_i s_{t-i}$ and the previously generated excitation e_{t-1}, where $e_t = s_t - y_t$.

LPCNet generates speech signals at an 8-bit resolution using μ-law companding. To shape the quantization noise and make it less perceptible a pre-emphasis filter $E(z) = 1 - \alpha z^{-1}$ is applied on the input speech (with $\alpha = 0.85$) and the inverse de-emphasis filter on the output. A major complexity saving comes from the insight that since s_{t-1}, y_t, and e_{t-1} are discrete, the contribution $\mathbf{v}_i^{(\cdot,\cdot)}$ of each possible value to the gates and state of GRU_A in Fig. 1 can be pre-computed. In addition, the contribution $\mathbf{g}^{(\cdot)}$ of the frame rate network to 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 then computed as (biases omitted for clarity)

$$\mathbf{u}_t = \sigma \left(\mathbf{W}_u \mathbf{h}_{t-1} + \mathbf{v}_{s_{t-1}}^{(u,s)} + \mathbf{v}_{y_t}^{(u,y)} + \mathbf{v}_{e_{t-1}}^{(u,e)} + \mathbf{g}^{(u)} \right)$$
$$\mathbf{r}_t = \sigma \left(\mathbf{W}_r \mathbf{h}_{t-1} + \mathbf{v}_{s_{t-1}}^{(r,s)} + \mathbf{v}_{y_t}^{(r,y)} + \mathbf{v}_{e_{t-1}}^{(r,e)} + \mathbf{g}^{(r)} \right) \qquad (2)$$
$$\widetilde{\mathbf{h}}_t = \tau \left(\mathbf{r}_t \circ (\mathbf{W}_h \mathbf{h}_{t-1}) + \mathbf{v}_{s_{t-1}}^{(h,s)} + \mathbf{v}_{y_t}^{(h,y)} + \mathbf{v}_{e_{t-1}}^{(h,e)} + \mathbf{g}^{(h)} \right)$$
$$\mathbf{h}_t = \mathbf{u}_t \circ \mathbf{h}_{t-1} + (1 - \mathbf{u}_t) \circ \widetilde{\mathbf{h}}_t$$
$$p(e_t) = \text{softmax} \left(\text{dual\_fc} \left(\text{GRU}_\text{B}(\mathbf{h}_t) \right) \right),$$

where $\sigma(x)$ is the sigmoid function, $\tau(x)$ is the hyperbolic tangent, $\circ$ denotes an element-wise vector multiply, and $\text{GRU}_\text{B}(\cdot)$ is a regular (non-sparse but smaller) GRU. The dual fully-connected ($\text{dual\_fc}(\mathbf{x})$) layer is defined as

$$\text{dual\_fc}(\mathbf{x}) = \mathbf{a}_1 \circ \tau(\mathbf{W}_1 \mathbf{x}) + \mathbf{a}_2 \circ \tau(\mathbf{W}_2 \mathbf{x}), \qquad (3)$$

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

The synthesized excitation sample e_t is obtained by sampling from the probability distribution $p(e_t)$ after lowering the temperature, i.e., decreasing the entropy of the distribution, of voiced frames as described in eq. (7) of [15]. To reduce complexity, 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
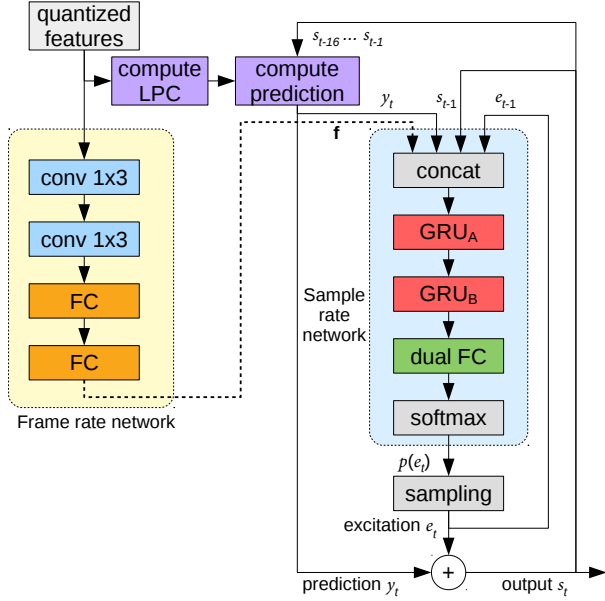
Figure 1: *Overview of the LPCNet model. The frame rate network (yellow) operates on frame-wise representing features and its output is held constant through each frame for the sample rate network (blue). The* compute prediction *block applies linear prediction to predict the sample at time t from the previous samples. Conversions between μ-law and linear are omitted for clarity. The de-emphasis filter is applied to the output s_t.*

those in $\mathbf{W}_r$ and $\mathbf{W}_u$, for an average of 10%. If N_A denotes the number of units in $\mathrm{GRU_A}$ the equivalent non-sparse number of units at a density d is $\sqrt{dN_A^2 + N_A}$.

In this work we use a model with 384 units in $\mathrm{GRU_A}$ (equivalent to 122 non-sparse units) and 16 units for $\mathrm{GRU_B}$, for a total of 72,000 weights in the sample rate network. This results in a total complexity of 3 GFLOPS. The complexity was also measured on different CPU architectures. On x86, real-time synthesis requires 20% of one 2.4 GHz Broadwell core (5x real-time). On ARMv8 (with Neon intrinsics), real-time LPCNet synthesis on a 2.5 GHz Snapdragon 845 (Google Pixel 3) requires 68% of one core (1.47x real-time). On the more recent 2.84 GHz Snapdragon 855 (Samsung Galaxy S10), real-time synthesis requires only 31% of one core (3.2x real-time).

4. Conditioning Features and Training

We mostly use the same set of conditioning features for both WaveNet and LPCNet. Those features are extracted from the Opus bit stream and represent the spectral shape and the pitch of the signal. There are two ways to compute the spectral envelope of the decoded audio:

1. Computing the spectrum on the decoded audio
2. Converting the LPCs into a spectrum

Both of those methods have significant drawbacks. For low bit rates, method 1 suffers from the quantization noise in the residual. On the other hand, while method 2 uses LPCs computed on clean speech, the fact that the reference encoder jointly optimizes LPC and LTP causes the frequency response of the LPC not to match the true spectrum of the signal. Because of

that, we use both methods and have two sets of spectral features, from each of which we compute 18 cepstral coefficients.

The five pitch gains from the LTP are summed to produce a single pitch gain feature. The sum represents the pitch gain for low frequencies, where it is most relevant. The LTP pitch period is used directly. The two sets of cepstral coefficients plus the two pitch parameters amount to a total of 38 features for LPCNet. The LPC parameters are computed from the decoded cepstrum rather than from the LPC in the Opus bit stream.

For LPCNet, best results are obtained when the whole network is trained on clean speech and then only the frame rate network is adapted with the decoded speech features. As reported in [26], adding noise to the network input signal can be beneficial to improve robustness to the training-inference mismatch caused by teacher forcing. For that reason, we add a small amount of Laplacian-distributed noise to the excitation inside the prediction loop, as shown in Fig. 3 of [27].

For WaveNet, the feature set described above does not result in synthesis with acceptable speech quality. Instead, only the cepstrum from method 2 is used and the model is trained directly on the decoded speech features.

To make both models more robust to variations in the input, we augment the training data. The signal level is varied over a 40 dB range and the frequency response is varied according to eq. (7) in [28].

5. Evaluation

The source code for the LPCNet model is available at `https://github.com/mozilla/LPCNet/` under a BSD license. The evaluation in this section is based on commit `2b64e3e`. The conditioning features are produced using a slightly modified Opus decoder found at commit `ec5bf39` of `https://github.com/xiph/opus/`.

5.1. Experimental Setup

The model is trained using four hours of 16 kHz-sampled speech (wideband) from the NTT Multi-Lingual Speech Database for Telephonometry (21 languages) [29]. We excluded all utterances from the speakers used in testing. Using the original data, we generated 14 hours of augmented speech data as described in Section 4.

We conducted a subjective listening test with a MUSHRA-inspired crowd-sourced methodology to evaluate the quality of neural synthesis of Opus parameters coded at 6 kb/s (average rate, since SILK is variable rate) in wideband mode. As an indication on the highest quality achievable with autoregressive neural synthesis, albeit at intractable complexity and latency, we included WaveNet synthesis. LPCNet synthesis represents a practically implementable system of today. We also compared with Opus [8] wideband (SILK mode) operating at 9 kb/s VBR[2]. We omitted a 3.5kHz LP-filtered original as anchor, which is the standard anchor in MUSHRA [30], as it likely would not be considered as the lowest quality. Instead, as a more appropriate low anchor we used Speex [31] operating as a 4 kb/s wideband vocoder (using the wideband quality setting at 0). The reference signal is sampled at 16 kHz.

In a first test (Set 1), we used eight utterances from two male and two female speakers. The utterances were from the NTT database used in training, but all utterances from the selected speakers for the test were excluded from the training

---

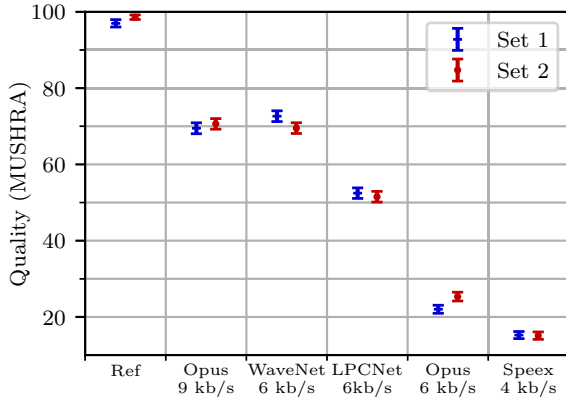[2]The lowest bit rate for which the encoder defaults to wideband if signal bandwidth is not specifically set.

Figure 2: *Listening test results for sets 1 and 2. The error bars indicate a 95% confidence interval.*

set. As reported in [13], mismatches between the training and testing databases can cause a significant difference in the output quality. We measure that impact in a second test (Set 2) on the same model, with eight utterances (one male and one female speaker) from the dataset used to create the Opus test vectors [32]. Each test included 100 listeners.

5.2. Results

We can see from the results in Fig. 2 that even though Opus produces an unacceptable wideband quality at 6 kb/s, both WaveNet and LPCNet provide sufficient improvements to make such a low rate usable. WaveNet synthesis from a 6 kb/s bit stream has a quality level comparable to Opus coded at 9 kb/s. LPCNet synthesis from the same bit stream yields a quality level that sits between Opus at 6 kb/s and 9 kb/s.

6. Conclusion

We have shown that neural synthesis can significantly improve the output of a low bit rate Opus bit stream. Previous speech coding efforts using neural synthesis were based on pure parametric coding, here we expand the scope to address also a waveform matching coder. Furthermore, when using the LPCNet architecture, real-time synthesis can be achieved even on a mobile device. This opens the door to improving other existing speech coders, such as AMR-WB [7], extending their life without breaking compatibility. Also, in this work, synthesis is performed using only frequency domain features, without directly using any temporal information from the decoded signal. In the future, even better quality may be achievable by using temporal processing with the time-domain decoded signal.

7. References

[1] T. Tremain, "Linear predictive coding systems," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, 1976, pp. 474–478.

[2] A. McCree, K. Truong, E. George, T. Barnwell, and V. Vishu, "A 2.4 kbit/s MELP coder candidate for the new U.S. federal standard," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, 1996, pp. 200 – 203.

[3] P. Hedelin, "A tone oriented voice excited vocoder," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 6, 1981, pp. 205–208.

[4] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, pp. 744–754, 1986.

[5] M. Schroeder and B. Atal, "Code-excited linear prediction (CELP): High-quality speech at very low bit rates," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 10, 1985, pp. 937–940.

[6] 3GPP, "TS 26.090: Adaptive multi-rate (AMR) speech codec; Transcoding functions," 2001.

[7] B. Bessette, R. Salami, R. Lefebvre, M. Jelinek, J. Rotola-Pukkila, J. Vainio, H. Mikkola, and K. Jarvinen, "The adaptive multi-rate wideband speech codec (AMR-WB)," *IEEE transactions on speech and audio processing*, vol. 10, no. 8, pp. 620–636, 2002.

[8] J.-M. Valin, K. Vos, and T. B. Terriberry, "Definition of the Opus Audio Codec," IETF RFC 6716, 2012, *https://tools.ietf.org/html/rfc6716*.

[9] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," *arXiv:1609.03499*, 2016.

[10] W. B. Kleijn, F. S. C. Lim, A. Luebs, J. Skoglund, F. Stimberg, Q. Wang, and T. C. Walters, "Wavenet based low rate speech coding," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018, pp. 676–680.

[11] C. Garbacea, A. v. den Oord, Y. Li, F. S. C. Lim, A. Luebs, O. Vinyals, and T. C. Walters, "Low bit-rate speech coding with VQ-VAE and a WaveNet decoder," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2019, pp. 735–739.

[12] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "SampleRNN: An unconditional end-to-end neural audio generation model," *arXiv:1612.07837*, 2016.

[13] J. Klejsa, P. Hedelin, C. Zhou, R. Fejgin, and L. Villemoes, "High-quality speech coding with sample RNN," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2019, pp. 7155–7159, *arXiv:1811.03021*.

[14] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," *arXiv:1802.08435*, 2018.

[15] J.-M. Valin and J. Skoglund, "LPCNet: Improving neural speech synthesis through linear prediction," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2019, *arXiv:1810.11846*.

[16] W3C, "WebRTC 1.0: Real-time communication between browsers," 2019, *https://www.w3.org/TR/webrtc/*.

[17] C. Holmberg, S. Håkansson, and G. Eriksson, "Web real-time communication use cases and requirements," IETF RFC 7478, Mar. 2015, *https://tools.ietf.org/html/rfc7478*.

[18] K. Vos, K. V. Sørensen, S. S. Jensen, and J.-M. Valin, "Voice coding with Opus," in *Proc. 135th Convention of the Audio Engineering Society*, 2013.

[19] J.-M. Valin, G. Maxwell, T. B. Terriberry, and K. Vos, "High-quality, low-delay music coding in the Opus codec," in *Proc. 135th Convention of the Audio Engineering Society*, 2013.

[20] J. Makhoul and M. Berouti, "Adaptive noise spectral shaping and entropy coding in predictive coding of speech," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 1, pp. 63–73, 1979.

[21] F. Itakura, "Line spectrum representation of linear predictor coefficients of speech signals," *The Journal of the Acoustical Society of America*, vol. 57, no. S1, pp. S35–S35, 1975.

[22] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *arXiv:1601.06759*, 2016.

[23] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixel-CNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications," *arXiv:1701.05517*, 2017.

[24] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Proc. Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, 2014, *arXiv:1409.1259*.

[25] B. S. Atal and S. L. Hanauer, "Speech analysis and synthesis by linear prediction of the speech wave," *The Journal of the Acoustical Society of America*, vol. 50, no. 2B, pp. 637–655, 1971.

[26] Z. Jin, A. Finkelstein, G. J. Mysore, and J. Lu, "FFTNet: A real-time speaker-dependent neural vocoder," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018, pp. 2251–2255.

[27] J.-M. Valin and J. Skoglund, "A real-time wideband neural vocoder at 1.6 kb/s using LPCNet," in *Proceedings of INTER-SPEECH*, 2019, pp. 3406–3410.

[28] J.-M. Valin, "A hybrid DSP/deep learning approach to real-time full-band speech enhancement," in *Proc. Multimedia Signal Processing Workshop (MMSP)*, 2018, *arXiv:1709.08243*.

[29] NTT Advanced Technology, *Multilingual Speech Database for Telephonometry*, 1994, CD-ROM.

[30] ITU-R, *Recommendation BS.1534-1: Method for the subjective assessment of intermediate quality level of coding systems*, International Telecommunications Union, 2001.

[31] J.-M. Valin, "The Speex codec manual," *Xiph.Org Foundation*, 2007.

[32] J.-M. Valin, K. Vos, and T. B. Terriberry, "Updates to the Opus Audio Codec," IETF RFC 8251, 2017, *https://tools.ietf.org/html/rfc6716*.