End-to-end LPCNet: A Neural Vocoder With Fully-Differentiable LPC Estimation

Krishna Subramani[*♭], *Jean-Marc Valin*[♮], *Umut Isik*[♮], *Paris Smaragdis*[♭♮], *Arvindh Krishnaswamy*[♮]

[♭] University of Illinois at Urbana-Champaign [♮] Amazon Web Services

ks51@illinois.edu, {jmvalin, umutisik, parsmara, arvindhk}@amazon.com

Abstract

Neural vocoders have recently demonstrated high quality speech synthesis, but typically require a high computational complexity. LPCNet was proposed as a way to reduce the complexity of neural synthesis by using linear prediction (LP) to assist an autoregressive model. At inference time, LPCNet relies on the LP coefficients being explicitly computed from the input acoustic features. That makes the design of LPCNet-based systems more complicated, while adding the constraint that the input features must represent a clean speech spectrum. We propose an end-to-end version of LPCNet that lifts these limitations by learning to infer the LP coefficients from the input features in the frame rate network . Results show that the proposed end-to-end approach equals or exceeds the quality of the original LPCNet model, but without explicit LP analysis. Our open-source[1] end-to-end model still benefits from LPCNet's low complexity, while allowing for any type of conditioning features.

Index Terms: neural speech synthesis, end-to-end optimization, linear prediction, LPCNet, WaveRNN

1. Introduction

Vocoder algorithms are used to synthesize intelligible speech from acoustic parameters. Over the past few years, vocoders based on deep neural networks (DNN) have vastly exceeded the capabilities of classical techniques such as linear predictive vocoders [1] and concatenative [2] synthesizers. This has resulted in significant quality improvements for text-to-speech (TTS) synthesis, speech compression, and other speech processing applications.

Many different types of neural vocoders have been proposed in the past, including autoregressive models such as WaveNet [3] and WaveRNN [4], but also GANs [5], and flow-based algorithms [6]. Most of these algorithms require either a GPU or a powerful CPU to synthesize speech in real time, limiting their use to servers or powerful devices. The LPCNet [7] vocoder, an improvement over WaveRNN, was recently introduced as a way of reducing the computational cost of neural synthesis. It uses linear prediction (LP) and the source-filter model of speech production [8] to simplify the task of the DNN model. The combination of signal processing with DNNs makes it possible to synthesize high quality speech with a complexity reducing the computational cost of ~ 3 GFLOPS, making LPCNet suitable for many existing phones.

LPCNet has so far been used in multiple applications, including compression [9], TTS [10] and voice conversion [11]. One limitation of the original LPCNet algorithm is that it requires explicit computation of the LP coefficients from the acoustic features at inference time. That requirement makes it impossible to use an arbitrary latent feature space, or even acoustic features that do not correspond to the clean speech being synthesized. LPCNet is thus unsuitable for a range of applications that includes end-to-end speech coding [12], end-to-end TTS [13], or cases where the input does not represent a full clean spectrum, such as bandwidth extension [14].

In this work, we propose an end-to-end differentiable LPCNet (Section 2) that avoids the explicit LPC computation. Instead, we learn LPC estimation from the input features by back-propagating the gradient from an improved loss function (Section 3). We evaluate the end-to-end LPCNet on a language-independent, speaker-independent task (Section 4). Results show that the proposed algorithm achieves the same quality as the original LPCNet (Section 5) while removing the need for the explicit LPC computation. That makes LPCNet suitable across a wide range of applications and devices (Section 6).

2. LPCNet Overview

WaveRNN [4] predicts a discrete probability distribution function (pdf) $P(s_t)$ for each sample s_t at time t from conditioning information, and the past samples up to s_{t-1}. It uses a gated recurrent unit (GRU), followed by linear layers and a softmax activation to output the distribution. The GRU uses block-sparse weight matrices to reduce the computational requirements.

LPCNet [7] builds upon WaveRNN, using linear prediction to simplify the DNN's task. LP coefficients a_i are explicitly computed from the input cepstral features, by first turning them back into a spectrum, from which it computes the autocorrelation, followed by the Levinson-Durbin algorithm. The LPCs are then used to compute a prediction p_t from the previous samples:

$$p_t = \sum_{i=1}^{M} a_i s_{t-i} \ , \tag{1}$$

where M is the prediction order. For LPCNet operating at 16 kHz, we use $M = 16$. The excitation, or residual, is then defined as $e_t = s_t - p_t$.

The main GRU ($\mathrm{GRU_A}$) can then use not only the past signal s_{t-1}, but also the past excitation e_{t-1} and the prediction for the current sample p_t. Similarly, the output estimates the distribution of the excitation $P(e_t)$.

To avoid WaveRNN's two-pass coarse-fine strategy to output the full 16-bit pdf, LPCNet uses the μ-law scale

$$U(x) = \mathrm{sgn}(x) \cdot \frac{U_{\max} \log(1 + \mu|x|)}{\log(1 + \mu)} \ , \tag{2}$$

where the μ-law range is $U_{\max} = 128$ and $\mu = 255$. Although μ-law values are typically represented as positive in the $[0, 256[$ range, in this paper we treat them as signed values within $[-128, 128[$ to simplify notation, with $U(0) = 0$.

[*] Work performed while at Amazon Web Services

[1] Source code available at `https://github.com/xiph/LPCNet/` in the `lpcnet_end2end` branch.

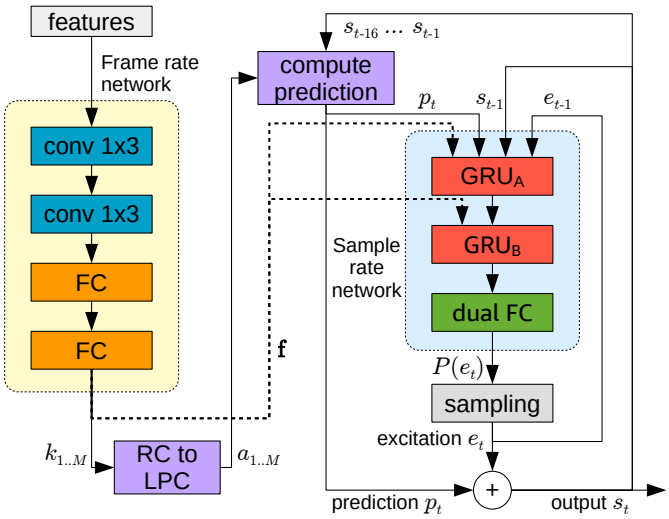

Figure 1: *Proposed end-to-end LPCNet architecture. The frame rate network operates on frames and computes the conditioning features* $\mathbf{f}$*, including the RCs* k_i *to be converted to LP coefficients* a_i*. The sample rate network predicts each sample, using the previous samples, as well as the excitation and prediction.*

The μ-law scale – which is also used in the original WaveNet proposal – makes the quantization noise independent of the signal amplitude. Furthermore, a pre-emphasis filter $E(z) = 1 - \alpha z^{-1}$ applied to the input avoids audible quantization noise from the 8-bit μ-law quantization. The inverse, de-emphasis filter $D(z) = \frac{1}{1-\alpha z^{-1}}$ is applied on the synthesis output. We use $\alpha = 0.85$.

A frame rate network learns frame conditioning features $\mathbf{f}$ from the cepstral features. A sample rate network predicts the output probability distribution for the excitation e_t when given the prediction p_t, the past excitation e_{t-1} and the past signal value s_{t-1}, along with the frame conditioning feature $\mathbf{f}$.

3. End-to-end LPCNet

We propose an end-to-end version of LPCNet where the LPC computation is differentiable and integrated into the algorithm rather than hardcoded based on a fixed set of cepstral features. Doing so makes it easier to use LPCNet, in a wider range of applications. The overview in Fig. 1 shows that the main difference with the original LPCNet is that the frame rate network is now used to compute the LPCs. To train a fully differentiable, end-to-end LPCNet model, three separate aspects have to be made differentiable:

1. LPC computation,
2. input embedding, and
3. loss function computation.

At inference time, only the LPC computation differs from the original LPCNet.

3.1. Learning to Compute LPCs

LP coefficients are very sensitive to error – and can easily become unstable – which is why they are never directly quantized or estimated. More robust representations include reflection coefficients (RC) [15], which guarantee stability as long as they lie in the $]-1, 1[$ interval, and line spectral frequencies [16], which need to follow an alternating ordering. Between those, we choose to estimate RCs, not only because the $]-1, 1[$ interval is easily enforced by the use of a $\tanh()$ activation, but also because it can be shown that the pre-tanh "logit" is equal (up to a scaling factor) to the log-area ratio [17] representation of RCs, which is known to be robust to error.

RCs can be converted to direct-form LP coefficients with the Levinson recursion [18]

$$a_j^{(j)} = \begin{cases} k_i, & \text{if } j = i \\ a_j^{(i-1)} + k_i a_{i-j}^{(i-1)}, & \text{otherwise} \end{cases} \tag{3}$$

where k_i are the RCs and $a_j^{(i)}$ are the j^{th} prediction coefficients of order i. Since both (1) and (3) are differentiable, the network can learn to compute RCs.

We directly use the first M elements of the frame rate network condition vector $\mathbf{f}$ (without increasing its dimensionality) as the RCs. This not only simplifies the design, but also ensures that the following sample rate network can take into account the estimated prediction coefficients. The complexity of both (3) and the cepstral-to-LPC conversion being replaced is negligible compared to the total inference complexity, therefore the proposed end-to-end system has essentially the same complexity as the original LPCNet.

The SFNet vocoder [19] also has its feature network learn RCs that match the spectral characteristics of the target synthesized speech. In the case of LPCNet, we want to directly learn the LP representation that optimizes the final loss function.

3.2. Differentiable Embedding Layer

The inputs p_t, s_{t-1}, e_{t-1} are quantized to 8-bit μ-law values, which are then used to learn an input sample embedding. We find that the embedding learns a set of non-linear functions that are applied to the input, improving synthesis quality compared to directly using the signal values as input to the GRU. However, μ-law quantization of the prediction and excitation prevents the gradient from backpropagating the loss all the way to the RC computation in the frame rate network. To make the embedding differentiable, we propose a simple interpolation scheme.

Let $\mathbf{v}_j$ be the j^{th} embedding vector of the embedding matrix and x be a real-valued (unquantized) μ-law sample value. We can compute the interpolated embedding $\mathbf{v}^{(i)}(x)$ as

$$f = x - \lfloor x \rfloor \tag{4}$$

$$\mathbf{v}^{(i)}(x) = (1 - f) \cdot \mathbf{v}_{\lfloor x \rfloor} + f \cdot \mathbf{v}_{\lfloor x \rfloor + 1} \,. \tag{5}$$

The approach allows a gradient to propagate through the fractional interpolation coefficient f.

At inference time, we avoid the (small) extra complexity of the interpolation and compute $\mathbf{v}(x) = \mathbf{v}_{\lfloor x \rceil}$, where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer.

3.3. Loss function

In the original LPCNet, the excitation can be pre-computed before the DNN training. In the proposed end-to-end LPCNet, the excitation is computed as part of the DNN architecture. The ground-truth target for the cross-entropy loss is thus no longer constant, raising two problems for our loss function. The first problem is that – as was the case for the embedding – quantizing the target μ-law excitation would not allow the gradient to backpropagate to the RC computation. We propose a similar solution to the interpolated embedding: an interpolated cross-entropy loss. Let $e_t^{(\mu)}$ be the real-valued μ-law excitation at

time t and let $\hat{P}(e_t^{(\mu)})$ be the discrete probability estimated for time t. Rather than rounding to the nearest integer and using the standard cross-entropy loss

$$\mathcal{L}_{\text{CE}} = \mathbb{E}\left[-\log \hat{P}\left(\left\lfloor e_t^{(\mu)} \right\rceil\right)\right] , \tag{6}$$

we interpolate the probability such that the loss becomes

$$f = e_t^{(\mu)} - \left\lfloor e_t^{(\mu)} \right\rfloor \tag{7}$$

$$\hat{P}^{(i)}\left(e_t^{(\mu)}\right) = (1-f)\hat{P}\left(\left\lfloor e_t^{(\mu)} \right\rfloor\right) + f\hat{P}\left(\left\lfloor e_t^{(\mu)} \right\rfloor + 1\right) \tag{8}$$

$$\mathcal{L}_{\text{ICE}} = \mathbb{E}\left[-\log\left(\hat{P}^{(i)}\left(e_t^{(\mu)}\right)\right)\right] , \tag{9}$$

where the expectation is taken over the distribution of the data over time. Again, the interpolation coefficient f helps the gradient backpropagate.

The second problem we have with our loss – and one that still applies to (9) – has to do with the non-linear nature of the μ-law scale. The μ-law spacing is wider for large excitation values and narrower for excitation values close to zero. It means that for the same linear uncertainty, the cross-entropy is larger when the excitation is large, *i.e.* when the predictor is *worse*.

We need to minimize a cross-entropy loss corresponding to the distribution of the linear-domain excitation samples. This could be done by simply dividing $\hat{P}^{(i)}(\cdot)$ by the linear step size corresponding to a difference of 1 in the μ-law value. Instead, since we are already interpolating to a continuous distribution, we divide by the derivative of the μ-law expansion function $U^{-1}(\cdot)$ to obtain a compensated loss:

$$\mathcal{L}_{\text{C}} = \mathbb{E}\left[-\log \frac{\hat{P}^{(i)}\left(e_t^{(\mu)}\right)}{\frac{d}{dx}U^{-1}(x)\big|_{e_t^{(\mu)}}}\right] . \tag{10}$$

Taking advantage of the fact that $U^{-1}(\cdot)$ is piecewise exponential and discarding constant terms in the loss, (10) simplifies to

$$\mathcal{L}_{\text{C}} = \mathcal{L}_{\text{ICE}} + \mathbb{E}\left[\frac{\left|e_t^{(\mu)}\right|\log(1+\mu)}{U_{\max}}\right] , \tag{11}$$

where the left-hand term $\mathcal{L}_{\text{ICE}}$ is our previously-derived interpolated cross-entropy loss in (9) and the right-hand term compensates for the μ-law scale with an L_1 loss on the μ-law companded excitation.

3.4. Regularization

We observe that using the compensated loss $\mathcal{L}_{\text{C}}$ alone often leads to the optimization process diverging. To stabilize the training, we add regularization to the LPC estimation. We consider the following three different regularization variants.

3.4.1. L_1 regularization

Our μ-law compensation in (11) already includes an L_1 loss, so the obvious regularizer is to artificially increase that compensation term using

$$\mathcal{L}_{L_1} = \gamma\mathbb{E}\left[\frac{\left|e_t^{(\mu)}\right|\log(1+\mu)}{U_{\max}}\right] . \tag{12}$$

We consider $\gamma = 1$, which effectively doubles the compensation term from (11) and is sufficient to stabilize the training. In effect, the regularization minimizes the prediction error in a similar way to the standard LPC analysis. Operating on the μ-law residual has the desirable property that the regularization applies almost equally to high- and low-amplitude signals.

3.4.2. Log-area ratio regularization

Another way of regularizing the LPCs is to directly attempt to match the "ground truth" LPCs used by the original LPCNet. Since those LPCs are computed on the target speech signal and are only needed at training time, the regularization does not impose additional constraints on the end-to-end LPCNet.

As a distance metric between the estimated RCs, k_i, and the ground truth RCs, $k^{(g)}$, we use the log-area ratio (LAR), which is both easy to compute and representative of the difference between the two filters:

$$\mathcal{L}_{\text{LAR}} = \mathbb{E}\left[\sum_i \left(\log\frac{1-k_i}{1+k_i} - \log\frac{1-k_i^{(g)}}{1+k_i^{(g)}}\right)^2\right] . \tag{13}$$

3.4.3. Log-area ratio matching

As a last option, we consider using only (13) to adapt the LPC estimation, while using the standard discrete cross-entropy (non-compensated) from (6). In that scenario, the LPCs attempt to match the ground truth LPCs without any regard for the output probability estimation process. The excitation is still computed based on the predicted LP coefficients.

4. Evaluation

We evaluate the end-to-end LPCNet on a speaker-independent, language-independent synthesis task where the inputs features are computed from a reference speech signal. We train all models on 205 hours of 16-kHz speech data from a combination of TTS datasets [20, 21, 22, 23, 24, 25, 26, 27, 28]. The data includes more than 900 speakers in 34 languages and dialects.

The training procedure includes noise augmentation in a similar way to the one described in [9]. To make the synthesis more robust to different microphone impulse responses, we include random spectral augmentation filtering [29]. Training sequences span 150 ms (using 10-ms frames), with 128 sequences per batch. All models are trained for 20 epochs (767k updates). For the end-to-end models, we add an extra epoch at the end of training, where the frame rate network weights are frozen so that the sample rate network can learn based on the final LPCs.

We evaluate the models on a combination of the PTDB-TUG speech corpus [30] and the NTT Multi-Lingual Speech Database for Telephonometry. From PTDB-TUG, we use all English speakers (10 male, 10 female) and randomly pick 200 concatenated pairs of sentences. For the NTT database, we select the American English and British English speakers (8 male, 8 female), which account for a total of 192 samples (12 samples per speaker). The training material does not include any speakers from the evaluation datasets.

We compare the original LPCNet with the three end-to-end variants from Sections 3.4.1, 3.4.2, and 3.4.3, denoted as L_1, LAR, and LAR/CE, respectively. For all models, we test with block sparse GRU_A sizes of 192, 384, and 640 units, using weight densities of 0.25, 0.1, and 0.15, respectively. The size of GRU_B is set to 32 units.

In addition to the original and end-to-end LPCNet models, we also evaluate the reference speech as an upper bound on quality, and we include the Speex 4 kb/s wideband vocoder [31] as a low anchor.

Table 1: *Log-spectral distance (LSD) between the end-to-end estimated LPCs and the ground truth LPCs over active frames of the evaluation set.*

Model	LSD (dB)		
GRU_A units	192	384	640
End-to-end L_1	3.58	3.64	3.64
End-to-end LAR	0.34	0.32	0.46
End-to-end LAR/CE	0.87	0.86	1.07

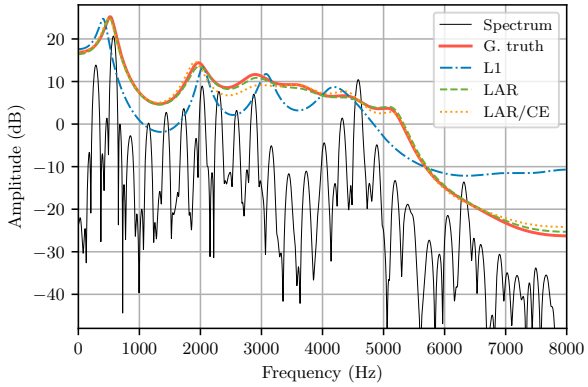

Figure 2: *Example spectrum for a pre-emphasized voiced frame of a female speaker. We show the LPC responses obtained with all models of size 384. The LAR and LAR/CE models follow the ground truth LPCs closely (LSD of 0.53 dB and 1.12 dB, respectively), but the L_1-regularized model has slightly different resonances and a higher "floor" (LSD of 7.17 dB).*

5. Results

We first compare the LPC estimation behavior of the different end-to-end variants. More specifically, we compare the response of the estimated LPCs with that of the explicit LPC analysis. Even though we refer to the latter as the *ground truth*, there is no assumption that it necessarily results in better synthesis quality than other LPC estimation methods. We measure the log-spectral distance between the ground truth LPCs and the end-to-end models on the active frames of all 384 test samples. The results in Table 1 show that both LAR-regularized models match the ground truth LPCs closely, whereas the L_1-regularized models show a moderate deviation from the ground truth LPCs. This is not unexpected, since the training does not attempt to match the ground truth explicitly, but rather minimize the residual. Fig. 2 provides an example of LPC responses for the different models.

We evaluate quality using mean opinion score (MOS) [32] using the crowdsourcing methodology described in P.808 [33]. Each speech sample is evaluated by 20 randomly-selected listeners. The results in Fig. 3 show that both the L_1- and LAR-regularized models perform similarly to the original LPCNet model, although they are significantly worse ($p < .05$) on some sizes. On the other hand, the LAR/CE model performs significantly worse than the baseline at all sizes and is also significantly worse than LAR on 2 sizes. That demonstrates the usefulness of the compensated, interpolated cross-entropy loss in the end-to-end training process. Further, our results demonstrate the our model's output quality is not directly linked to how closely the estimated LPCs match the ground truth LPCs.

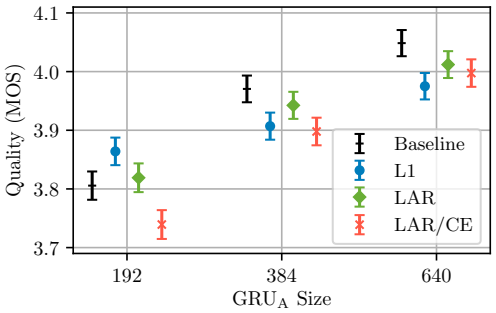

Figure 3: *Results from the MOS quality evaluation, including the 95% confidence intervals. The reference speech has a MOS of* 4.21 ± 0.02 *and the Speex anchor has a MOS of* 2.76 ± 0.04.

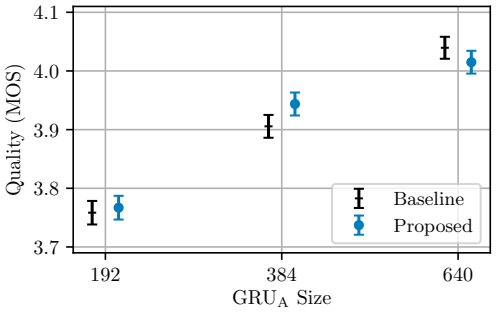

Figure 4: *Results from the MOS quality evaluation, including the 95% confidence intervals. The reference speech has a MOS of* 4.21 ± 0.02 *and the Speex anchor has a MOS of* 2.56 ± 0.03. *It is clear that the proposed method performs as well or better than the baseline, while providing the extra advantage of being fully-differentiable.*

To close the slight quality gap found in the results of Fig. 3, we propose to use both L_1 and LAR regularization. We evaluate the new samples using the same methodology, with 20 randomly-selected listeners per sample. The results in Fig. 4 show that the proposed model is significantly better than the baseline model at size 384 and is statistically tied for the other two sizes.

6. Conclusion

We have proposed an end-to-end version of LPCNet where the LPCs no longer need to be explicitly computed. We also demonstrated a compensated, interpolated cross-entropy loss that improves the performance of the trained end-to-end models. The end-to-end LPCNet achieves equal or better synthesis quality compared to the original LPCNet, while removing the constraint of having interpretable acoustic features for the conditioning. The resulting vocoder architecture makes it easier to use LPCNet, for a broader range of applications. We believe that this work can be extended to bring the benefits of linear prediction to other ML-based audio tasks, such as time-domain speech enhancement systems. It is an open question whether the proposed technique could also be applied to non-autoregressive neural synthesis techniques, such as GAN- and flow-based models.

7. References

[1] J. Markel and A. Gray, "A linear prediction vocoder simulation based upon the autocorrelation method," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 22, no. 2, pp. 124–134, 1974.

[2] A.J. Hunt and A.W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1996, vol. 1, pp. 373–376.

[3] 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 preprint arXiv:1609.03499*, 2016.

[4] 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," in *Proc. International Conference on Machine Learning (ICML)*, 2018, pp. 2410–2419.

[5] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *Proc. ICLR*, 2019.

[6] R. Prenger, R. Valle, and B. Catanzaro, "WaveGlow: A flow-based generative network for speech synthesis," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[7] J.-M. Valin and J. Skoglund, "LPCNet: Improving neural speech synthesis through linear prediction," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5891–5895.

[8] G. Fant, *Acoustic theory of speech production*, Walter de Gruyter, 1960.

[9] J.-M. Valin and J. Skoglund, "A real-time wideband neural vocoder at 1.6 kb/s using LPCNet," in *Proc. INTERSPEECH*, 2019.

[10] Z. Kons, S. Shechtman, A. Sorin, C. Rabinovitz, and R. Hoory, "High quality, lightweight and adaptable TTS using LPCNet," in *Proc. INTERSPEECH*, 2019.

[11] S. Zhao, H. Wang, T.H. Nguyen, and B. Ma, "Towards natural and controllable cross-lingual voice conversion based on neural TTS model and phonetic posteriorgram," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[12] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "SoundStream: An end-to-end neural audio codec," 2021.

[13] J. Donahue, S. Dieleman, M. Bińkowski, E. Elsen, and K. Simonyan, "End-to-end adversarial text-to-speech," in *Proc. International Conference on Learning Representations (ICLR)*, 2021.

[14] A. Gupta, B. Shillingford, Y. Assael, and T.C. Walters, "Speech bandwidth extension with WaveNet," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2019.

[15] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 561–580, 1975.

[16] F. Itakura, "Line spectrum representation of linear predictive coefficients of speech signals," *Journal of the Acoustical Society of America*, vol. 57, no. S1, 1975.

[17] R. Viswanathan and J. Makhoul, "Quantization properties of transmission parameters in linear predictive systems," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 3, pp. 309–321, 1975.

[18] N. Levinson, "The Wiener RMS error criterion in filter design and prediction," *Journal of Mathematical Physics*, vol. 25, pp. 261—278, 1947.

[19] A. Rao and P.K. Ghosh, "SFNet: A computationally efficient source filter model based neural speech synthesis," *IEEE Signal Processing Letters*, vol. 27, pp. 1170–1174, 2020.

[20] I. Demirsahin, O. Kjartansson, A. Gutkin, and C. Rivera, "Open-source Multi-speaker Corpora of the English Accents in the British Isles," in *Proc. LREC*, 2020.

[21] O. Kjartansson, A. Gutkin, A. Butryna, I. Demirsahin, and C. Rivera, "Open-Source High Quality Speech Datasets for Basque, Catalan and Galician," in *Proc. SLTU and CCURL*, 2020.

[22] K. Sodimana, K. Pipatsrisawat, L. Ha, M. Jansche, O. Kjartansson, P. De Silva, and S. Sarin, "A Step-by-Step Process for Building TTS Voices Using Open Source Data and Framework for Bangla, Javanese, Khmer, Nepali, Sinhala, and Sundanese," in *Proc. SLTU*, 2018.

[23] A. Guevara-Rukoz, I. Demirsahin, F. He, S.-H. C. Chu, S. Sarin, K. Pipatsrisawat, A. Gutkin, A. Butryna, and O. Kjartansson, "Crowdsourcing Latin American Spanish for Low-Resource Text-to-Speech," in *Proc. LREC*, 2020.

[24] F. He, S.-H. C. Chu, O. Kjartansson, C. Rivera, A. Katanova, A. Gutkin, I. Demirsahin, C. Johny, M. Jansche, S. Sarin, and K. Pipatsrisawat, "Open-source Multi-speaker Speech Corpora for Building Gujarati, Kannada, Malayalam, Marathi, Tamil and Telugu Speech Synthesis Systems," in *Proc. LREC*, 2020.

[25] Y. M. Oo, T. Wattanavekin, C. Li, P. De Silva, S. Sarin, K. Pipatsrisawat, M. Jansche, O. Kjartansson, and A. Gutkin, "Burmese Speech Corpus, Finite-State Text Normalization and Pronunciation Grammars with an Application to Text-to-Speech," in *Proc. LREC*, 2020.

[26] D. van Niekerk, C. van Heerden, M. Davel, N. Kleynhans, O. Kjartansson, M. Jansche, and L. Ha, "Rapid development of TTS corpora for four South African languages," in *Proc. Interspeech*, 2017.

[27] A. Gutkin, I. Demirşahin, O. Kjartansson, C. Rivera, and K. Túbọ̀sún, "Developing an Open-Source Corpus of Yoruba Speech," in *Proc. INTERSPEECH*, 2020.

[28] E. Bakhturina, V. Lavrukhin, B. Ginsburg, and Y. Zhang, "Hi-Fi Multi-Speaker English TTS Dataset," *arXiv preprint arXiv:2104.01497*, 2021.

[29] J.-M. Valin, "A hybrid DSP/deep learning approach to real-time full-band speech enhancement," in *Proceedings of IEEE Multimedia Signal Processing (MMSP) Workshop*, 2018.

[30] G. Pirker, M. Wohlmayr, S. Petrik, and F. Pernkopf, "A pitch tracking corpus with evaluation on multipitch tracking scenario.," in *Proc. INTERSPEECH*, 2011, pp. 1509–1512.

[31] J.-M. Valin, "The Speex codec manual," 2007.

[32] ITU-T, *Recommendation P.800: Methods for subjective determination of transmission quality*, 1996.

[33] ITU-T, *Recommendation P.808: Subjective evaluation of speech quality with a crowdsourcing approach*, 2018.