Interspeech 2022
18-22 September 2022, Incheon, Korea

Bunched LPCNet2: Efficient Neural Vocoders Covering Devices from Cloud to Edge

Sangjun Park[1], *Kihyun Choo*[1], *Joohyung Lee*[1], *Anton V. Porov*[2], *Konstantin Osipov*[2], *June Sig Sung*[3]

[1]Samsung Research, Samsung Electronics, Republic of Korea
[2]PDMI RAS, Russia
[3]Mobile eXperience Business, Samsung Electronics, Republic of Korea
{sj0.park, khchoo, jooooh.lee} @samsung.com

Abstract

Text-to-Speech (TTS) services that run on edge devices have many advantages compared to cloud TTS, e.g., latency and privacy issues. However, neural vocoders with a low complexity and small model footprint inevitably generate annoying sounds. This study proposes a Bunched LPCNet2, an improved LPCNet architecture that provides highly efficient performance in high-quality for cloud servers and in a low-complexity for low-resource edge devices. Single logistic distribution achieves computational efficiency, and insightful tricks reduce the model footprint while maintaining speech quality. A DualRate architecture, which generates a lower sampling rate from a prosody model, is also proposed to reduce maintenance costs. The experiments demonstrate that Bunched LPCNet2 generates satisfactory speech quality with a model footprint of 1.1MB while operating faster than real-time on a RPi 3B. Our audio samples are available at https://srtts.github.io/bunchedLPCNet2.

Index Terms: text-to-speech, neural vocoder, LPCNet, bunched LPCNet, edge computing

1. Introduction

Nowadays, as many types of edge devices emerge, people utilize text-to-speech (TTS) services in their daily life without device-constraints. Although most TTS systems now have been launched on cloud servers, running on edge devices resolves significant concerns, such as latency, privacy, and internet connectivity issues. Note that edge devices include high-resource devices, such as smartphones, and low-resource devices, such as smart watches, wireless earphones, and home assistant devices. To deploy TTS services on these devices optimizing the performance within a given resource is crucial. Additionally, expanding the service capability while maintaining the architecture is worthwhile instead of developing new architectures for specific devices. For this purpose, FBWave [1] proposed a flow-based scalable vocoder architecture to easily control the computational cost. However, it demonstrated an insufficient speech quality in cloud servers, even in high-quality mode. This study introduces a vocoder family that can provide a wide coverage on computational cost constraints from cloud to low-resource edge devices.

Many neural vocoders have been proposed since WaveNet, which predicts a waveform directly, was presented [2]. Early neural vocoders are based on a computationally expensive auto-regressive (AR) manner [2, 3], impractical for real-time services. Inferring in parallel on GPUs have also been attempted [4, 5, 6], but few edge devices have a GPU. On the other hand, LPCNet [7] is a light-weight vocoder based on AR architecture that is suitable for on-device inference. To the best of our knowledge, LPCNet is still one of the best candidates that can efficiently generate speech on a CPU, even with a high quality. Many LPCNet variants have been introduced: iLPCNet [8] and Full-Band LPCNet [9] were studied to improve speech quality; and Bunched LPCNet [10], Gaussian LPCNet [11], and Sub-band LPCNet [12] were studied to reduce complexity without sacrificing speech quality. However, the variants are unsatisfactory for low-resource edge devices in that they target high-end devices, such as smart phones. A quality drop is also observed by shrinking the network capacity for low-resource devices.

In this paper, we introduces Bunched LPCNet2, which is an improved Bunched LPCNet that addresses the problems arising when covering a wide range of devices, from cloud servers to wearable devices such as smart watches and AR/VR glasses. Our contributions include three methods:

1. Single logistic output layer that extends the coverage of Bunched LPCNet to low-resource constraints.
2. Dual-rate LPCNet for computational and maintenance cost reduction.
3. Insightful tricks that significantly reduce the model footprint.

We present a brief overview of Bunched LPCNet in Section 2, followed by an in-depth description of the proposed methods in Section 3. Section 4 summarizes the evaluation results in terms of computational cost and speech quality. Finally, Section 5 concludes the paper.

2. Bunched LPCNet

In our previous study [10], Sample Bunching was proposed as depicted in Figure 1. It allows LPCNet to generate multiple samples per GRU inference. Bit Bunching was also proposed to reduce the computations in the DualFC and softmax layers by splitting the output bits in two: the higher 7 bits for coarse prediction and lower 4 bits for fine correction.

Bunching techniques are sufficiently efficient for application in mobile devices, e.g., a smartphone. However, several limitations exist for further reducing computations for low-resource devices. (1) Sample Bunching reduces the computations of GRUs by $1/S$ times. This indicates that as the bunch size S increases, the amount of computational reduction is saturated, especially where $S > 4$. (2) Reducing the number of GRU_A units with Bunching techniques significantly degrades the speech quality. Note that GRU_A is the most computationally expensive layer in LPCNet. (3) The model footprint becomes large because of the embedding tables corresponding to the feedback samples.

10.21437/Interspeech.2022-310

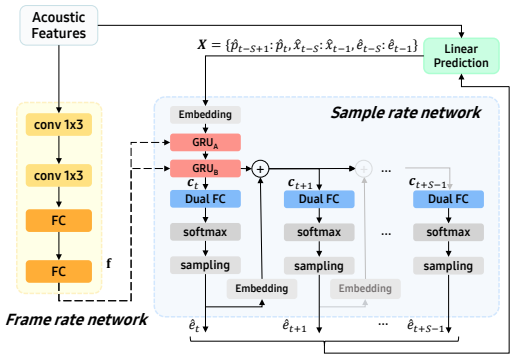


Figure 1: *Bunched LPCNet overview*

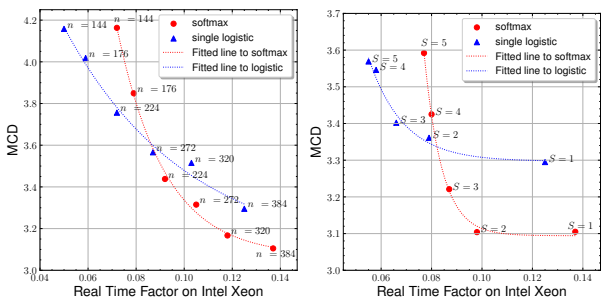
To overcome these problems, we improved the Bunched LPCNet architecture with the three methods described in Section 3.

3. Proposed Methods

3.1. Single logistic output layer

Auto-regressive vocoders generate waveforms by sampling from the probability density function with non-parametric or parametric distributions. Non-parametric methods, e.g., a softmax output layer, can model complex distributions and show a good performance in modeling waveforms [2, 7], but the computations in the output layer and sampling process from a categorical distribution are expensive by a large output dimension; for example, 256 for an 8-bits μ-law quantization [13]. Additionally, its speech quality is significantly degraded when the model has insufficient capacity.

Accordingly, we attempted to employ parametric distributions based on previous studies [14, 15], which used a mixture of logistics (MoL) as an output distribution, and determined that a logistic output layer, particularly with single mixture, is a good alternative in terms of efficiency. To demonstrate the efficiency of the logistic layer in complexity reduction, we compare the mel-cepstral distances (MCD) [16] of single logistic (SL) to the softmax output layer with the smaller bunch size S and GRU_A units n_a in Figure 2. Note that $S = 1$ and $n_a = 384$ correspond to the original LPCNet. As the model capacity decreases, the SL layers work more efficiently than the softmax layers. We believe that a non-parametric loss could lead the network to predict complex distributions beyond the model capacity. That is, it could cause an under-fitting problem when the



(a) *on GRU_A units (S=1)* (b) *on bunch size (n_a=384)*

Figure 2: *MCD plots of softmax and single logistic output layer*

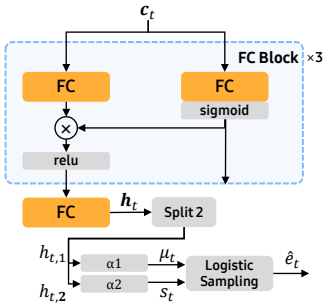


Figure 3: *Network architecture for single logistic output layer*

model capacity is too small. In contrast, a unimodal parametric distribution simplifies an objective and lowers the training difficulty by reducing the burden on the network. It can mitigate the quality reduction despite an insufficient capacity.

Figure 3 presents our network architecture for the SL layer. The FC block comprises fully-connected (FC) layers with 16 units, and the output dimension of a final FC layer is 2. A location parameter μ_t and scale parameter s_t are obtained from the custom activation functions $\alpha 1$ and $\alpha 2$ as follows.

$$\mu_t = \alpha 1(h_{t,1}) = \tanh\left(h_{t,1}/u\right) \quad (1)$$

$$s_t = \alpha 2(h_{t,2}) = \exp\left(\tanh\left(h_{t,2}\right) \cdot v - w\right) \quad (2)$$

u, v, and w help maintain the training stable, and are set empirically, to 64, 16 and 6, respectively. The model is trained to maximize the discretized logistic likelihood [17] on normalized 16-bit samples from -1 to +1. Finally, $\hat{e}_t$ is sampled from the logistic distribution, as defined in (3).

$$\hat{e}_t = \mu_t + T \cdot s_t \cdot \ln(\epsilon/(1-\epsilon)), \quad \epsilon \sim \text{Uniform}(0, 1) \quad (3)$$

where T denotes a temperature parameter. When $\hat{e}_t$ is fed to the embedding tables, the 8-bit μ-law quantization is applied.

Employing the deep and narrow layers, the proposed architecture demonstrates a better performance while satisfying a lower complexity than the softmax layer. Moreover, the sampling process of the SL layer is undoubtedly simple and computationally cheap using only scalar operations.

3.2. DualRate LPCNet

For low-resource devices, a more compact model would be required. However, a model with low capacity inevitably generates annoying sounds, e.g., noisy or muffled sounds. Reducing the sampling rate could be a solution of this problem.

By the way, a prosody model of a TTS system requires continuous maintenance, even after deployed, so that the pronunciation errors or unnatural prosody for given texts can be fixed. In other words, employing an additional prosody model for a low sampling rate vocoder leads to a maintenance cost increase. For this reason, we propose a DualRate LPCNet, which is an architecture that provides two sampling rates (24 kHz and 16 kHz) from one prosody model, as depicted in Figure 4. Note that x_i and F_i denote the i kHz waveform and acoustic features extracted from x_i, respectively.

The feature conversion block computes F'_{16}, which is the input of 16 kHz LPCNet, to remove unnecessary information from F_{24} and obtain higher mutual information with a target x_{16}. The acoustic features F_{24}, which comprise 20 cepstral coefficients, a pitch period, and a pitch correlation, are converted as follows.

Table 1: *System Configurations*

Systems	Output Layer	S	n_a	n_e	T	Sampling rate	Target device
B-LPCNet[10]	Softmax with Bit Bunching	4	384	128	0.75	24 kHz	High-end
B-LPCNet2-L	Softmax	1	384	1	0.75	24 kHz	Cloud
B-LPCNet2-R	Single logistic	2	224	1	0.75	24 kHz	High-end
B-LPCNet2-S	Single logistic	5	176	1	0.65	24 kHz	Low-end
B-LPCNet2-S16	Single logistic	5	176	1	0.65	16 kHz	Low-end

- The 20 cepstral coefficients are extracted with 20 CELT bands from the Opus codec [18]. The 20 CELT bands cover up to a 12 kHz bandwidth, and the last two bands cover from 8 kHz to 12 kHz. The two unnecessary bands are removed by sequentially computing the IDCT, low-pass filter, and DCT. Finally, the 18 cepstral coefficients that cover up to an 8 kHz bandwidth are obtainable.
- The pitch period is multiplied by the ratio of sampling rates ($\frac{16k}{24k} = \frac{2}{3}$).
- The pitch correlation is used as is.

Using the 16 kHz LPCNet trained with the converted F'_{16}, we can save on both computational and maintenance costs. Similarly, the TTS systems for given devices can be easily deployed by choosing suitable LPCNet models with various complexities.

3.3. Optimizing model footprint

Although Sample Bunching has an efficient architecture for reducing the computational complexity, it increases the number of model parameters by the embedding tables corresponding to the feedback samples X, where $\boldsymbol{X} = \{\hat{p}_{t-S+1} : \hat{p}_t, \hat{x}_{t-S} : \hat{x}_{t-1}, \hat{e}_{t-S} : \hat{e}_{t-1}\}$, as depicted in Figure 1. As mentioned in [10], $E_{x \in \boldsymbol{X}}$ is converted into a pre-computed lookup table E'_x obtained from the multiplication of E_x and U_x, the input weight matrix in GRU_A, to replace matrix-vector multiplication operations with vector-vector addition operations. Specifically, E_x with a $[2^8, n_e]$ shape is converted to E'_x with a $[2^8, 3 \cdot n_a]$ shape, where n_e and n_a denote the embedding dimension of E_x and number of GRU_A units, respectively. The total number of parameters in E'_x is $2^8 \cdot 3 \cdot n_a \cdot (3 \cdot S)$. With the default hyper-parameters of Bunched LPCNet, $n_e = 128$ and $n_a = 384$, their parameters account for 70% of the model, even when $S = 1$.

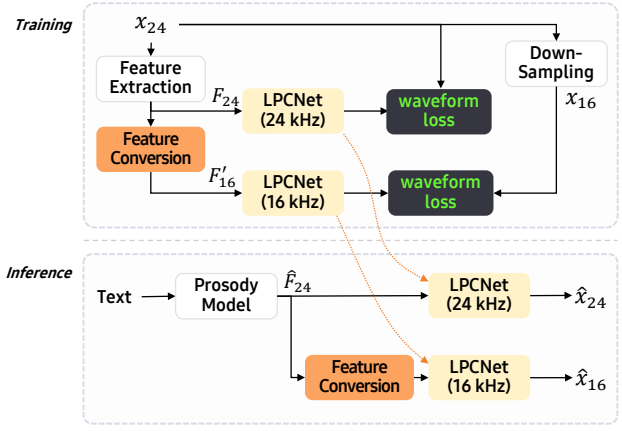


Figure 4: *Training / inference procedure of DualRate LPCNet.*

Therefore, we focus on reducing the embedding table size.

To optimize the model footprint, we attempted to feedback the scalar values into GRU_A. The first method worth considering is to feedback the scalar values directly without the embedding table. However, this wastes the model capacity to transform the scalar values into the desired representation from the network. To feedback desired values into the network without wasting model capacity, a nonlinear mapping function from a quantized value to a continuous value would be useful, which is the embedding table with $n_e = 1$. This could degrade the speech quality compared to when $n_e = 128$. However, we determined that it has a small effect on the performance, whereas the size of E_x and U_x can be significantly reduced.

Note that n_e does not affect the size of the dumped model owing to E'_x with the fixed shape $[2^8, 3 \cdot n_a]$ being independent from n_e. Thus, we dumped E_x with the $[2^8, n_e]$ shape and U_x with the $[n_e, 3 \cdot n_a]$ shape separately into a model instead of E'_x (hereafter referred to as separated format). Note that it works only when $2^8 \cdot n_e + n_e \cdot 3 \cdot n_a < 2^8 \cdot 3 \cdot n_a$. For example, when $n_e = 128$, n_a must be greater than 85. The greater n_a and the less n_e achieve the higher reduction ratio. The total number of parameters is reduced to $(2^8 \cdot n_e + n_e \cdot 3 \cdot n_a) \cdot (3 \cdot S)$ without performance degradation. E'_x can be restored during inference engine initialization. The separated format facilitates an extremely small model footprint: 70% reduction for $S = 1$ and 82% reduction for $S = 2$ when $n_e = 1$ and $n_a = 384$.

4. Experimental Results

4.1. Experimental environment

The several combinations of hyper parameters were chosen empirically to satisfy various device specifications as summarized in Table 1, compared with Bunched LPCNet. We attempted to control the complexity of the models using S and n_a and determined an output layer to efficiently work for a given model capacity. For example, *B-LPCNet2-R* was chosen because the SL layer is more efficient than the softmax layer where $S = 2$ and $n_a = 224$ in Figure 2. In addition, the temperature parameter T was adjusted to suppress the noise generated by the low-complexity models. Finally, the *B-LPCNet2-L*, *B-LPCNet2-R*, and *B-LPCNet2-S* target cloud, high-end, and low-end devices, respectively, and *B-LPCNet2-S16* was prepared for low-resource devices that need to lower the sampling rate. The other hyperparameters were set identically to [10].

A Tacotron variant architecture [19] was employed as a prosody model to evaluate the speech quality in the TTS pipeline. The systems were trained using two datasets: one of a professional English male speaker (17-hours with 10,000 utterances) and another of a professional English female speaker (15-hours with 7,612 utterances). Each vocoder model and prosody model were trained with a single speaker. Sixty utterances were used as a test set, and one percent of the remaining

Table 2: *MOS with 95% confidence intervals and real time factor on each CPU architecture*

Systems	MOS	RTF	
		Intel Xeon	RPi
Original(24kHz)	4.51 ± 0.04	-	-
Original(16kHz)	4.12 ± 0.05	-	-
B-LPCNet	3.95 ± 0.06	0.072	2.394
B-LPCNet2-L	4.08 ± 0.06	0.137	5.538
B-LPCNet2-R	3.99 ± 0.06	0.051	1.657
B-LPCNet2-S	3.90 ± 0.06	0.030	0.720
B-LPCNet2-S16	3.81 ± 0.06	0.021	0.507
WORLD	-	0.075	1.808

utterances were used as the validation set for training.

4.2. Performance

For the quality evaluation, we conducted mean opinion score (MOS) tests [20] on the Amazon Mechanical Turk platform with 300 people and 120 unseen test utterances. For the complexity evaluation, we measured the RTF (Real Time Factor) on two devices: 1) an AWS c5.4xlarge instance (Intel Xeon Platinum 8124M CPU @ 3.00GHz) with Ubuntu 18.04 - representing a cloud device and 2) a Raspberry Pi (RPi) 3B v1.2 (BCM2837 @ 1.20 GHz) with Tizen 6.5 - representing an edge device. Our implementation was optimized using SIMD (Single Instruction Multiple Data) with single thread for each architecture. To demonstrate that our systems are as efficient as conventional vocoders, we also measured the complexity of the WORLD vocoder [21][1]. Table 2 presents the evaluation results.

The *B-LPCNet2* systems with an SL layer demonstrated a highly efficient performance. The *B-LPCNet2-L* generated sufficiently high fidelity speech for cloud TTS. The *B-LPCNet2-R* achieved a lower complexity than the *B-LPCNet* and even the WORLD vocoder. The *B-LPCNet2-S* and *B-LPCNet2-S16* works 7.7x and 10.9x faster than the *B-LPCNet2-L*, respectively, on the RPi while maintaining a satisfactory speech quality. Employing them with a lightweight prosody model, a real-time TTS system is deployable on low-resource edge devices.

4.3. Model footprint

To investigate the efficiency in terms of the model footprint, we compared the MOS results to the model size. Notably, the separated format in Section 3.3 is applied to all evaluated systems, and the model size included file headers less than 1 KB. Table 3 summarize the results.

The systems with $n_e = 128$, including *B-LPCNet*, has a relatively large number of parameters. This results from the model size of Bunched LPCNet highly depending on the bunch size, embedding dimension, and GRU_A units, as mentioned in Section 3.3. In contrast, the *B-LPCNet2* architectures show significantly small footprints of approximately 1.1 MB. Indeed, without the separated format, the size of the *B-LPCNet* is 15.4 MB. Compared to the original Bunched LPCNet, this means that the *B-LPCNet2-R* generates higher speech quality with only 7% of the parameters.

[1]An open source code at `https://github.com/mmorise/World` was used

Table 3: *MOS with 95% confidence intervals and the model footprint for different embedding dimensions*

Systems	MOS	Model Size
B-LPCNet	3.95 ± 0.06	10.141 MB
B-LPCNet2-L	4.08 ± 0.06	1.136 MB
└ $n_e = 128$	4.07 ± 0.06	3.496 MB
B-LPCNet2-R	3.99 ± 0.06	1.135 MB
└ $n_e = 128$	3.97 ± 0.06	3.834 MB
B-LPCNet2-S	3.90 ± 0.06	1.099 MB
└ $n_e = 128$	3.87 ± 0.06	4.898 MB
B-LPCNet2-S16	3.81 ± 0.06	1.071 MB

To confirm the effect of the embedding dimension n_e on speech quality, we also compared the MOS results for $n_e = 1$ and $n_e = 128$. We expected that the speech quality would degrade considerably, but the quality difference is negligible.

Figure 5 illustrates the trained embedding values of *B-LPCNet2-S*. Note that the horizontal lines at both ends arose from our optimization trick that maps the sparse indices to the adjacent dense index, preventing access to untrained weights in inference. Each line has a different scale, slope, symmetrical point, and nonlinearity. This suggests that the trained embedding table with $n_e = 1$ controls the nonlinear transforms with desirable properties from the network. Evidently, this is more outstanding than directly feedbacking the scalar value.

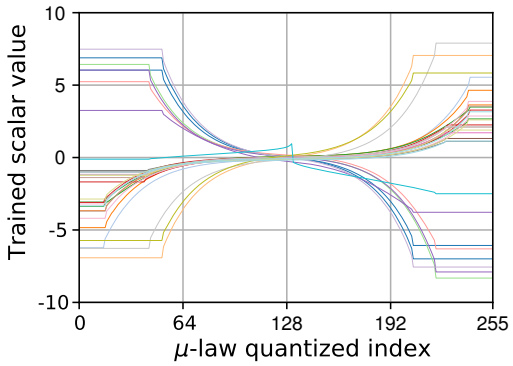


Figure 5: *The plot of trained embedding value with* $n_e = 1$. *(A total of 30 lines = 2 speakers · 3 feedback samples · 5 Sample Bunching)*

5. Conclusion

We introduced Bunched LPCNet2, which is a highly efficient neural vocoder architecture, and some presets to provide high-fidelity real-time TTS service to satisfy various device specifications, as confirmed by the MOS tests and RTF evaluations. We also investigated a simple technique that can significantly reduce the model size without degrading the speech quality. Compared with our previous study [10], *B-LPCNet2-R* achieved better speech quality and lower complexity with a model size of only 1.1 MB. In terms of quality, complexity, and model footprint, Bunched LPCNet2 would be the best candidate for low-resource edge devices, such as smart watches, wireless earphones, and home assistant devices.

6. References

[1] B. Wu, Q. He, P. Zhang, T. Koehler, K. Keutzer, and P. Vajda, "Fb-wave: Efficient and scalable neural vocoders for streaming text-to-speech on the edge," *arXiv preprint arXiv:2011.12985*, 2020.

[2] A. v. d. 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.

[3] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *Proc. of International Conference on Machine Learning (ICML)*, 2018, pp. 2410–2419.

[4] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3617–3621.

[5] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," in *Proc. of Advances in Neural Information Processing Systems*, 2019.

[6] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Proc. of Advances in Neural Information Processing Systems*, 2020.

[7] J.-M. Valin and J. Skoglund, "LPCNet: Improving Neural Speech Synthesis through Linear Prediction," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5891–5895.

[8] M.-J. Hwang, E. Song, R. Yamamoto, F. Soong, and H.-G. Kang, "Improving lpcnet-based text-to-speech with linear prediction-structured mixture density network," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7219–7223.

[9] K. Matsubara, T. Okamoto, R. Takashima, T. Takiguchi, T. Toda, Y. Shiga, and H. Kawai, "Full-band lpcnet: A real-time neural vocoder for 48 khz audio with a cpu," *IEEE Access*, vol. 9, pp. 94 923–94 933, 2021.

[10] R. Vipperla, S. Park, K. Choo, S. Ishtiaq, K. Min, S. Bhattacharya, A. Mehrotra, A. G. C. P. Ramos, and N. D. Lane, "Bunched lpcnet : Vocoder for low-cost neural text-to-speech systems," in *Proc. of INTERSPEECH*, 2020, pp. 3565–3569.

[11] V. Popov, M. Kudinov, and T. Sadekova, "Gaussian lpcnet for multisample speech synthesis," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6204–6208.

[12] Y. Cui, X. Wang, L. He, and F. K. Soong, "An efficient subband linear prediction for lpcnet-based neural synthesis." in *Proc. of INTERSPEECH*, 2020, pp. 3555–3559.

[13] *ITU-T. Recommendation G. 711.*, Pulse Code Modulation (PCM) of voice frequencies, 1988.

[14] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, R. A. Saurous, Y. Agiomvrgiannakis, and Y. Wu, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4779–4783.

[15] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis, "Parallel WaveNet: Fast high-fidelity speech synthesis," in *Proc. of International Conference on Machine Learning (ICML)*, 2018, pp. 3918–3926.

[16] R. Kubichek, "Mel-cepstral distance measure for objective speech quality assessment," in *Proc. of IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, vol. 1, 1993, pp. 125–128.

[17] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Improving the pixelcnn with discretized logistic mixture likelihood and other modifications," in *Proc. of International Conference on Learning Representations (ICLR)*, 2017.

[18] J.-M. Valin, G. Maxwell, T. B. Terriberry, and K. Vos, "High-quality low-delay music coding in the opus codec," *Audio Eng. Soc. Conv. 135*, p. 8942, 2013.

[19] N. Ellinas, G. Vamvoukakis, K. Markopoulos, A. Chalamandaris, G. Maniati, P. Kakoulidis, S. Raptis, J. S. Sung, H. Park, and P. Tsiakoulis, "High quality streaming speech synthesis with low, sentence-length-independent latency," in *Proc. of INTERSPEECH*, 2020, pp. 2022–2026.

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

[21] M. Morise, F. Yokomori, and K. Ozawa, "World: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Transactions on Information and Systems*, vol. E99.D, no. 7, pp. 1877–1884, 2016.