NOLACE: IMPROVING LOW-COMPLEXITY SPEECH CODEC ENHANCEMENT THROUGH ADAPTIVE TEMPORAL SHAPING

Jan Büthe, Ahmed Mustafa, Jean-Marc Valin, Karim Helwani, Michael M. Goodwin

Amazon Web Services
Palo Alto, USA
{jbuethe, ahdmust, jmvalin, helwk, mmg}@amazon.com

ABSTRACT

Speech codec enhancement methods are designed to remove distortions added by speech codecs. While classical methods are very low in complexity and add zero delay, their effectiveness is rather limited. Compared to that, DNN-based methods deliver higher quality but they are typically high in complexity and/or require delay. The recently proposed Linear Adaptive Coding Enhancer (LACE) addresses this problem by combining DNNs with classical long-term/short-term postfiltering resulting in a causal low-complexity model. A short-coming of the LACE model is, however, that quality quickly saturates when the model size is scaled up. To mitigate this problem, we propose a novel adatpive temporal shaping module that adds high temporal resolution to the LACE model resulting in the Non-Linear Adaptive Coding Enhancer (NoLACE). We adapt NoLACE to enhance the Opus codec and show that NoLACE significantly outperforms both the Opus baseline and an enlarged LACE model at 6, 9 and 12 kb/s. We also show that LACE and NoLACE are well-behaved when used with an ASR system.

Index Terms— speech enhancement, speech coding, Opus, DDSP

1. INTRODUCTION

Degradation of speech through coding is a common problem in real-time communication scenarios where bandwidth is often limited and the speech codec therefore needs to operate below the transparency threshold. Improving such degraded speech has been a long-standing problem and a large variety of both classical and DNN based solutions have been proposed to address this issue [1, 2, 3, 4, 5, 6].

While classical methods are very light-weight, DNN based methods tend to be either very complex or they require additional delay and often both is the case. This makes it difficult to deploy these methods for real-time applications on low-end devices like smart phones and it also makes it challenging to integrate them into switching codecs like Opus or EVS in which the speech codec is just one of many embedded modes.

The Linear Adaptive Coding Enhancer (LACE) [7] addresses these issues by combining the classical approach with a DNN which is used to calculate filter coefficients for long-term and short-term filters on a 5-ms-frame basis. These are then applied to the degraded signal in the classical way to enhance its harmonic structure and spectral envelope. The result is a very lightweight model that requires only 100 MFLOPS and is fully linear in the signal path. The model is furthermore fully causal on 20-ms frames and it is trained to be phase preserving. This allows for direct integration into existing communication codecs, where phase preservation ensures that seamless mode switching, as e.g. implemented in Opus or EVS, is maintained.

LACE has been shown [7] to significantly improve the Opus codec at 6, 9 and 12 kb/s and it provided about 60 % of the MOS improvement of the non-causal LPCNet resynthesis method [3], which requires 25 ms lookahead and comes with a complexity 3 GFLOPS.

A major drawback of the LACE model is, however, that quality quickly saturates when scaling up the model size, which essentially restricts it to be a very-low complexity tool with medium quality gain.

In this paper, we identify the low temporal resolution of LACE as the main cause for quality saturation. To mitigate this, we design an adaptive temporal shaping module, which calculates sample-wise gains on a frame basis using as input a feature vector and a temporal envelope of the signal to be shaped. We add the shaping module as a third custom DSP module to the LACE model and since the temporal shaping module also adds non-linear processing to the signal path we refer to the resulting model as Non-Linear Adaptive Coding Enhancer (NoLACE).

We adapt NoLACE as a multi-bitrate enhancer for the Opus speech coding mode[1] and verify in a P.808 listening test that NoLACE significantly outperforms a LACE model of equivalent size at 6, 9 and 12 kb/s. The results show furthermore, that NoLACE at 12 kb/s scores close to the clean signal and that NoLACE at 6 kb/s achieves 92% of the MOS improvement of the non-causal LPCNet resynthesis method.

As a second evaluation metric we test ASR performance by measuring the word error rates (WER) for Opus in combination with LACE, NoLACE and the LPCNet resynthesis method using the large SpeechBrain conformer model. The results show that LACE and NoLACE significantly improve WER at the lowest bitrate while the LPCNet resynthesis method leads to further degradation compared to the Opus baseline. At higher bitrates both LACE and NoLACE deliver WER close to the Opus baseline, which quickly converge to the WER of the clean signal.

Although we adapt NoLACE to enhance the Opus speech coding mode it can generally be used with any speech codec that provides explicit pitch information. Furthermore, the custom differentiable DSP (DDSP) modules used to build LACE and NoLACE can likely be used for other signal processing tasks. An implementation of the NoLACE model including a general implementation of the DDSP modules is available in the opus repository.[2]

The tested NoLACE model has a complexity of $\approx$ 620 MFLOPS which requires only a small fraction of the compute capability available on common laptop or smart phone CPUs. Compared to fully

[1] Demo samples are available at https://282fd5fa7.github.io/NoLACE
[2] https://gitlab.xiph.org/xiph/opus/-/tree/icassp2024

neural codecs [8, 9, 10, 11, 12, 13, 14], the approach of enhancing an existing codec also has the practical advantage of maintaining backward compatibility, leaving an inexpensive decoding option for low-end devices like microcontrollers.

2. PROPOSED MODEL

2.1. Notation

Throughout this paper we denote the clean signal by $x(t)$, the coded signal by $y(t)$ and the enhanced signal by $\hat{y}(t)$. Furthermore, we assume all these signals to be pre-emphasised with a factor 0.85. Furthermore, n always denotes the sub-frame index of the Opus linear-predictive mode which operates on 5-ms subframes.

2.2. Background: LACE

The LACE model consists of two parts, a signal processing module and a feature encoder. The signal processing module implements two consecutive adaptive comb-filters, which make explicit use of a pitch lag p_n, followed by an adaptive convolution. The purpose of the feature encoder is to combine information from several input features into a latent feature vector φ_n, which is used in the signal processing module to compute filter coefficients on a 5-ms-frame basis. To avoid discontinuities, filter coefficients are interpolated on the first half of the 5-ms frames. Except for this input-independent interpolation, the system is essentially constant on individual frames, which restricts the temporal resolution 200 Hz.

For enhancing Opus, the input features are a mix of clean-signal features computed and quantized by the encoder (spectrum, pitch lag, ltp coefficients), noisy-signal features computed from the signal y (cepstrum, auto-correlation) and bitrate information extracted at the decoder. For a detailed description of the input features we refer to [7].

2.3. NoLACE

The NoLACE model has the same basic design as LACE. It consists of a feature encoder identical to the LACE feature encoder and a signal processing module displayed in Figure 1. The first part of the signal processing module follows the design of LACE, having two consecutive adaptive comb-filter modules (AdaComb) and one adaptive convolution module (AdaConv). What follows next is an iteration of a select-shape-mix procedure that is build around the adaptive temporal shaping module (AdaShape): the AdaConv1 module produces two output channels (select), one passing through the adaptive temporal shaping module AdaShape1 (shape) and one bypassing it. The two channels are then mixed together by the AdaConv2 module which, producing two output channels, instantaneously performs the selection operation for the next iteration. The reasoning behind this is the following: the initial AdaConv module implements a spectral shaping and as such selects the frequency components that are to be shaped by the AdaShape module. Since the AdaShape module is non-linear in the signal path, the selection also serves as aliasing control. The bypass channel carries complementary signal parts that are required to reconstruct the signal in the final mixing operation. The select-shape-mix procedure is carried out three times with AdaConv4 producing the final enhanced signal $\hat{y}(t)$.

A second addition to the signal processing module are additional feature transformations that are used to filter the latent feature vector while handing it down from layer to layer. While the main quality

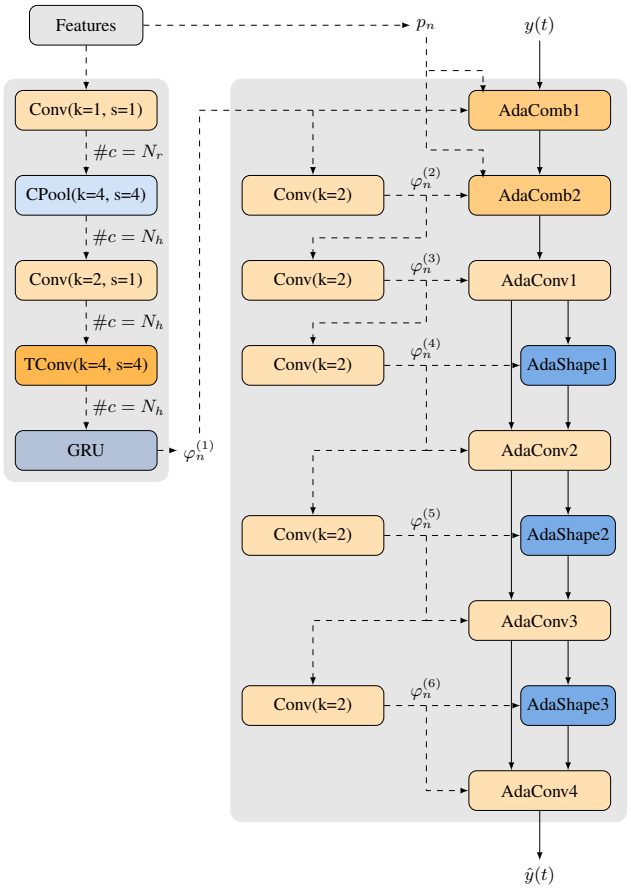

Fig. 1. High-level overview of the NoLACE model. The feature encoder which transforms the input features into a latent feature vector $\varphi_n^{(1)}$ is depicted on the left and the number of channels are indicated with $\#c =$. The signal processing unit on the right applies first a series of two comb-filtering and spectral shaping operation before entering a select-shape-mix iteration involving the proposed adaptive temporal shaping modules AdaShape

improvement for NoLACE comes from the temporal shaping modules, this additional filtering has been found to provide an additional small but significant quality improvement.

NoLACE and LACE share the same hyper parameters N_r (the reduced feature dimension) and N_h (the number of hidden channels, the GRU size and the dimension of the latent feature vectors $\varphi_n^{(k)}$). In fact, LACE can be recovered from NoLACE as the first output channel of the AdaConv1 module in Figure 1 when setting $\varphi_n^{(3)} := \varphi_n^{(2)} := \varphi_n^{(1)}$. While the LACE was trained with $N_r = 96$ and $N_h = 128$ in [7], we choose $N_r = 96$ and $N_h = 256$ for NoLACE. We similarly increase N_h for the LACE comparison model in this paper.

2.4. The Adaptive Temporal Shaping Module

The adaptive temporal shaping module (AdaShape) modifies the input signal by multiplying each sample by an individual gain. These sample-wise gains are calculated on a frame basis from a temporal envelope, derived from the input signal as the average absolute values on 4-sample blocks, and a feature vector φ_n. To reduce the

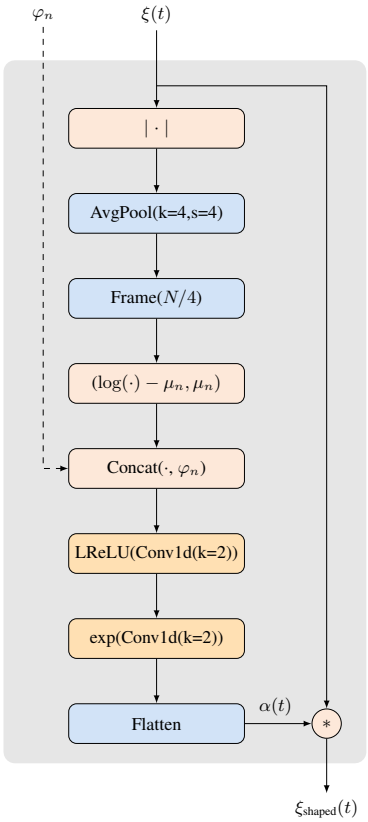

Fig. 2. Adaptive temporal shaping module. Shapes are given in channels last format, N denotes the frame size and μ denotes the frame-wise mean value.

dynamic range, modelling is done in log-domain. To this end the temporal envelope is transformed to log domain and the frame-mean value μ_n is subtracted. The resulting zero-mean envelope is then concatenated with the mean value μ_n and the given feature vector φ_n which are then passed through a two-layer convolutional network to derive the sample-wise gains in log domain as illustrated in Figure 2.

2.5. Multi-Channel Adaptive Convolutions

The select-shape-mix iteration requires adaptive convolutions with two channels. Since the AdaConv module was only defined for a single input and output channel in [7], we extend the definition to an arbitrary number of input and output channels. This is done in analogy to regular 1d convolutions but kernel normalization and gain computation require some attention. In the original definition the single channel impulse response is calculated from a feature vector φ_n as product of a shape

$$\kappa_n = \frac{W_\kappa \varphi_n + b_\kappa}{\|W_\kappa \varphi_n + b_\kappa\|_2} \tag{1}$$

and a gain

$$g_n = \exp(\alpha \tanh(W_g \varphi_n + b_g)) \tag{2}$$

to derive the impulse response for a single frame

$$h_n(\tau) = g_n \, \kappa_n(\tau), \tag{3}$$

where τ denotes the filter tap index. The impulse responses are then interpolated on the first half of the frame to provide smooth transitions.

We extend this definition first to m_1 input channels and a single output channel by defining it as the sum of m_1 adaptive convolutions, where the j-th kernel shape is given by

$$\kappa_n^{(j)} = \frac{W_\kappa^{(j)} \varphi_n + b_\kappa^{(j)}}{\sum_{i=1}^{m} \|W_\kappa^{(i)} \varphi_n + b_\kappa^{(i)}\|_2} \tag{4}$$

and where the kernel gain is given by (2) for all m_1 adaptive convolutions, i.e. we jointly normalize over all input channels and use a shared gain. An adaptive convolution with m_1 input channels and m_2 output channels is defined by concatenating m_2 adaptive convolutions with m_1 input channels and a single output channel along the channel dimension.

3. TRAINING

We trained on 165 hours of clean speech sampled at 16 kHz which are collected from multiple high-quality TTS datasets [15, 16, 17, 18, 19, 20, 21, 22, 23] containing more than 900 speakers in 34 languages and dialects. The input signals y were generated using a patched version of libopus that restricts Opus to linear-predictive mode and wideband encoding. Furthermore, the encoder is modified to randomly switch encoding parameters bitrate, complexity and packet_loss_percent every 249-th frame.[3]

3.1. Model Pre-Training

In a first step, NoLACE is pre-trained using the same combination of regression losses, $10\,\mathcal{L}_{\text{phase}} + 2\,\mathcal{L}_{\text{env}} + \mathcal{L}_{\text{spec}}$, from [7]. Pre-training is carried out for 50 epochs using the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, a sequence length of 0.5 seconds, a batch size of 256 and a learning rate decay factor of 2.5×10^{-5}.

3.2. Adversarial Training

For adversarial training we use spectrogram discriminators at multiple resolutions as proposed in [24]. We follow the setup in [25] and use 6 STFT discriminators D_k but apply a few modifications: First, each discriminator takes as input a log-magnitude spectrogram calculated from size-2^{k+5} STFTs with 75% overlap. By abuse of notation we still write $D_k(x)$ or $D_k(\hat{y})$, treating the log-magnitude STFT transform as part of the discriminator. We furthermore apply strides along the frequency axis to keep the frequency range of the receptive fields constant. This has been found to increase the ability of discriminators with high frequency resolution to detect inter-harmonic noise. Finally, we concatenate a two-dimensional frequency positional sine-cosine embedding to the input channels of every 2d-convolutional layer.

We train NoLACE as a least-squares GAN [26]. First we note that the coded input signal y depends only on the clean signal x and the encoder parameters π_{enc}. With this we define the adversarial part of the training loss for NoLACE as

$$\mathcal{L}_{\text{adv}}(x, \hat{y}) = \frac{1}{6} \sum_{k=1}^{6} E_{x, \pi_{\text{enc}}}[(1 - D_k(\hat{y}))^2] + \mathcal{L}_{\text{feat}}(D_k, x, \hat{y}), \tag{5}$$

where $\mathcal{L}_{\text{feat}}$ denotes the standard feature matching loss, i.e. the mean of the L^1 losses of hidden layer outputs for x and $\hat{y}$.

[3] https://gitlab.xiph.org/xiph/opus/-/tree/exp-neural-silk-enhancement

For regularization and to maintain phase preservation we also add the following combination of pre-training losses

$$\mathcal{L}_{\text{reg}} := \frac{60}{155}\mathcal{L}_{\text{env}} + \frac{30}{155}\mathcal{L}_{\text{phase}} + \frac{3}{155}\mathcal{L}_{\text{spec}}, \tag{6}$$

whence the final training loss for NoLACE is given by

$$\mathcal{L}_{\text{NoLACE}}(x, \hat{y}) = \mathcal{L}_{\text{adv}}(x, \hat{y}) + \mathcal{L}_{\text{reg}}(x, \hat{y}). \tag{7}$$

Simultaneously, the discriminators are trained to minimize the losses

$$\mathcal{L}_{D_k}(x, \hat{y}) = E_{x,\pi_{\text{enc}}}[D_k(\hat{y})^2 + (1 - D_k(x))^2]. \tag{8}$$

Adversarial training is carried out for another 50 epochs with a fixed learning rate of 10^{-4} and a batch size of 64, which corresponds to roughly 930 K training steps. We used the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for both NoLACE and the discriminators.

4. EVALUATION

4.1. Listening Test

We evaluated the quality of NoLACE using 192 clean English speech clips from the NTT Multi-Lingual Speech Database for Telephonometry, which was not included in the training data. Using the crowd-sourcing methodolgy from ITU-R P.808[27], we tested Opus, Opus + LACE and Opus + NoLACE at 6, 9, and 12 kb/s.

For a fair comparison we trained both LACE and NoLACE with $N_r = 96$ and $N_h = 256$. This leads to a complexity of 280 MFLOPS with 900 K parameters for LACE (roughly a 3x increase) and a complexity of 620 MFLOPS with 1.8 M parameters for NoLACE. No adversarial training was performed for the LACE model since it was found to degrade quality.

We furthermore included the LPCNet resynthesis method from [3] at 6 kb/s as an additional test point. The method adds 25 ms delay to the decoder and is therefore not a realistic replacement option for LACE or NoLACE so it rather serves as an interesting reference point.

The results show that NoLACE consistently outperforms the large LACE model. Furhtermore, NoLACE achieves roughly 92% of the MOS improvement of LPCNet resynthesis[3]. LACE, on the other hand, achieves about 66% of MOS improvement, which is 6 pp higher than the results reported for the smaller LACE model in [7]. This indicates that quality saturates indeed quickly when enlarging the model size.

4.2. ASR testing

We evaluated the impact of Opus and the three enhancement methods LACE, NoLACE and LPCNet resynthesis on ASR performance of the large SpeechBrain [28] conformer ASR model[4] using the native clean speech test set of the LibriSpeech ASR corpus [29]. The results in Table 1 show that Opus coding has a significant impact on ASR performance at very low bitrates, increasing WER by roughtly one pp at 6 kb/s. While the resynthesis method further increases WER by approx. 0.2 pp at this bitrate, both LACE and NoLACE enhancement leads to an improvement of approx. 0.6 resp 0.5 pp indicating that LACE and NoLACE make up for about half the errors introduced by Opus coding.

For higher bitrates, Opus, LACE and NoLACE quickly converge to the clean speech WER indicating that LACE and NoLACE can be used with an ASR system without having to retrain.

[4] https://huggingface.co/speechbrain/asr-conformer-transformerlm-librispeech

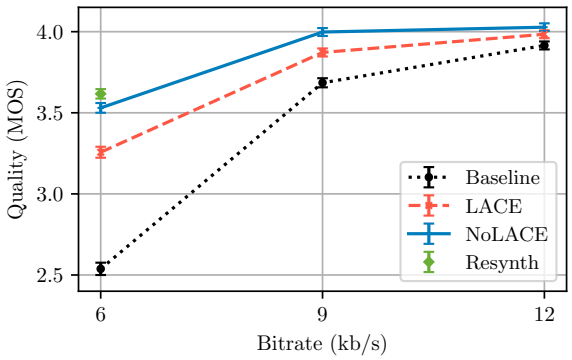

Fig. 3. P.808 results. The clean signal has a MOS of 4.06 ± 0.025. LACE consistently outperforms the baseline and NoLACE consistently outperforms LACE at all bitrates. At 6 kb/s, NoLACE achieves 92% of the MOS improvement of the LPCNet resynthesis method which requires 25ms delay and 5x the complexity of NoLACE.

condition	6 kb/s	9 kb/s	12 kb/s	20 kb/s
clean	2.01%	2.01%	2.01%	2.01%
Opus	3.08%	2.15%	2.07%	2.03%
Opus + LACE	2.46%	2.13%	2.05%	2.03%
Opus + NoLACE	2.56%	2.18%	2.06%	2.02%
Opus + resynthesis	3.26%	-	-	-

Table 1. Word error rates for Opus in combination with different enhancement methods. At the lowest bitrate WER is significantly increased for Opus condition. LACE and NoLACE significantly reduce WER for this bitrate while the LPCNet resynthesis method further increases it. At higher bitrates WER for LACE and NoLACE quickly reduce and are close to Opus WER.

5. CONCLUSION

In this paper we identified low temporal resolution as the main bottleneck of the LACE model for scaling to higher quality. We introduced the adaptive temporal shaping module (AdaShape), used it to design the Non-Linear Adaptive Coding Enhancer (NoLACE) and demonstrateed in a P.808 listening test that NoLACE consistently outperforms LACE. Furthermore, we conducted an ASR test which showed that ASR performance is maintained, and at low bitrates even improved, when adding LACE or NoLACE. The model is causal and phase-preserving and can be implemented with insignificant complexity overhead on common smart phone CPUs, which allows for direct integration into codecs with dedicated speech-coding mode. We believe such a low-complexity enhancement algorithm will be most useful for enhancing the quality of existing classical speech codecs while maintaining compatibility.

6. ACKNOWLEDGMENT

We would like to thank Minho Jin for assisting with the ASR test.

7. REFERENCES

[1] J.-H. Chen and A. Gersho, "Adaptive Postfiltering for Quality Enhancement of Coded Speech," *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 1, pp. 59–71, 1995.

[2] Z. Zhao, H. Liu, and T. Fingscheidt, "Convolutional Neural Networks to Enhance Coded Speech," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 4, pp. 663–678, 2019.

[3] J. Skoglund and J.-M. Valin, "Improving Opus Low Bit Rate Quality with Neural Speech Synthesis," in *Proc. INTERSPEECH*, 2019.

[4] K. Gupta, S. Korse, B. Edler, and G. Fuchs, "A DNN Based Post-Filter to Enhance the Quality of Coded Speech in MDCT Domain," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 836–840.

[5] S. Korse, K. Gupta, and G. Fuchs, "Enhancement of Coded Speech Using a Mask-Based Post-Filter," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6764–6768.

[6] S. Korse, N. Pia, K. Gupta, and G. Fuchs, "PostGAN: A GAN-Based Post-Processor to Enhance the Quality of Coded Speech," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 831–835.

[7] J. Büthe, J.-M. Valin, and A. Mustafa, "LACE: A light-weight, causal model for enhancing coded speech through adaptive convolutions," in *Proc. Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2023.

[8] J.-M. Valin and J. Skoglund, "A Real-Time Wideband Neural Vocoder at 1.6kb/s Using LPCNet," in *Proc. INTERSPEECH*, 2019, pp. 3406–3410.

[9] W. B. Kleijn, A. Storus, M. Chinen, T. Denton, F. S. C. Lim, A. Luebs, J. Skoglund, and H. Yeh, "Generative Speech Coding with Predictive Variance Regularization," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 6478–6482.

[10] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "SoundStream: An End-to-End Neural Audio Codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 495–507, 2022.

[11] Nicola P., Kishan G., Srikanth K., Markus M., and Guillaume F., "NESC: Robust Neural End-2-End Speech Coding with GANs," in *Proc. INTERSPEECH*, 2022.

[12] T. Jenrungrot, M. Chinen, W. B. Kleijn, J. Skoglund, Z. Borsos, N. Zeghidour, and M. Tagliasacchi, "LMCodec: A Low Bitrate Speech Codec With Causal Transformer Models," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

[13] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High Fidelity Neural Audio Compression," 2022, arXiv:2210.13438.

[14] Y.-C. Wu, I. D. Gebru, D. Marković, and A. Richard, "Audiodec: An Open-Source Streaming High-Fidelity Neural Audio Codec," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] E. Bakhturina, V. Lavrukhin, B. Ginsburg, and Y. Zhang, "Hi-Fi Multi-Speaker English TTS Dataset," in *Proc. INTERSPEECH*, 2021, pp. 2776–2780.

[24] W. Jang, D. C. Y. Lim, J. Yoon, B. Kim, and J. Kim, "UnivNet: A Neural Vocoder with Multi-Resolution Spectrogram Discriminators for High-Fidelity Waveform Generation," in *Proc. INTERSPEECH*, 2021.

[25] A. Mustafa, J.-M. Valin, J. Büthe, P. Smaragdis, and M. Goodwin, "Framewise WaveGAN: High speed adversarial vocoder in time domain with very low computational complexity," in *ICASSP 2023*, 2023.

[26] X. Mao, Q. Li, H. Xie, R. Lau, W. Zhen, and S. Smolley, "Least Squares Generative Adversarial Networks," 10 2017, pp. 2813–2821.

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

[28] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J.-C. Chou, S.-L. Yeh, S.-W. Fu, C.-F. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R. De Mori, and Y. Bengio, "SpeechBrain: A General-Purpose Speech Toolkit," 2021, arXiv:2106.04624.

[29] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 04 2015, pp. 5206–5210.