Scalable and Efficient Neural Speech Coding: A Hybrid Design

Kai Zhen, *Student Member, IEEE*, Jongmo Sung, Mi Suk Lee, Seungkwon Beack, and Minje Kim, *Senior Member, IEEE*

***Abstract*—We present a scalable and efficient neural waveform coding system for speech compression. We formulate the speech coding problem as an autoencoding task, where a convolutional neural network (CNN) performs encoding and decoding as a neural waveform codec (NWC) during its feedforward routine. The proposed NWC also defines quantization and entropy coding as a trainable module, so the coding artifacts and bitrate control are handled during the optimization process. We achieve efficiency by introducing compact model components to NWC, such as gated residual networks and depthwise separable convolution. Furthermore, the proposed models are with a scalable architecture, cross-module residual learning (CMRL), to cover a wide range of bitrates. To this end, we employ the residual coding concept to concatenate multiple NWC autoencoding modules, where each NWC module performs residual coding to restore any reconstruction loss that its preceding modules have created. CMRL can scale down to cover lower bitrates as well, for which it employs linear predictive coding (LPC) module as its first autoencoder. The hybrid design integrates LPC and NWC by redefining LPC's quantization as a differentiable process, making the system training an end-to-end manner. The decoder of proposed system is with either one NWC (0.12 million parameters) in low to medium bitrate ranges (12 to 20 kbps) or two NWCs in the high bitrate (32 kbps). Although the decoding complexity is not yet as low as that of conventional speech codecs, it is significantly reduced from that of other neural speech coders, such as a WaveNet-based vocoder. For wide-band speech coding quality, our system yields comparable or superior performance to AMR-WB and Opus on TIMIT test utterances at low and medium bitrates. The proposed system can scale up to higher bitrates to achieve near transparent performance.**

***Index Terms*—Neural speech coding, waveform coding, representation learning, model complexity.**

I. Introduction

SPEECH coding can be implemented as an encoder-decoder system, whose goal is to compress input speech signals into the compact bitstream (encoder) and then to reconstruct the original speech from the code with the least possible quality degradation. Speech coding facilitates telecommunication and saves data storage among many other applications. There is a typical trade-off a speech codec must handle: the more the system reduces the amount of bits per second (bitrate), the worse the perceptual similarity between the original and recovered signals is likely to be perceived. In addition, the speech coding systems are often required to maintain an affordable computational complexity when the hardware resource is at a premium. For decades, speech coding has been intensively studied yielding various standardized codecs that can be categorized into two types: the vocoders and waveform codecs. A vocoder, also referred to as parametric speech coding, distills a set of physiologically salient features, such as the spectral envelope (equivalent to vocal tract responses including the contribution from mouth shape, tongue position and nasal cavity), fundamental frequencies, and gain (voicing level), from which the decoder *synthesizes* the speech. Typically, a vocoder operates at 3 kbps or below with high computational efficiency, but the synthesized speech quality is usually limited and does not scale up to higher bitrates [1]–[3]. On the other hand, a waveform codec aims to accurately reconstruct the input speech signal, which features up-to-transparent quality in a high bitrate range [4]. AMR-WB [5], for instance, can be seen as a hybrid waveform codec, because it employs speech modeling as in many other waveform codecs [6]–[8]. EVS [9], a recently standardized 3GPP voice and audio codec, has noticeably optimized frame error robustness, yielding a much-enhanced frame error concealment performance against than AMR-WB [10]. Similar to EVS, Opus, a waveform codec at its core, can also be applied to both speech and audio signals where it uses the LPC-based SILK algorithm for the speech-oriented model [11] and scales up to 510 kbps for transparent audio streaming and archiving.

Under the notion of unsupervised speech representation learning, deep neural network (DNN)-based codecs have revitalized the speech coding problem and provided different perspectives [12], [13]. The major motivation of employing neural networks to speech coding is twofold: to fill the performance gap between vocoders and waveform codecs towards a near-transparent speech synthesis quality; to use its trainable encoder

Manuscript received March 26, 2021; revised June 28, 2021 and September 9, 2021; accepted November 8, 2021. Date of publication November 19, 2021; date of current version December 20, 2021. This work was supported by the Institute for Information and Communications Technology Promotion (IITP) funded by the Korea government (MSIT) under Grant 2017-0-00072 (Development of Audio/Video Coding, and Light Field Media Fundamental Technologies for Ultra Realistic Tera-Media). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Tom Backstrom. *(Corresponding author: Minje Kim.)*

Kai Zhen is with the Department of Computer Science and Cognitive Science Program, Indiana University, Bloomington, IN 47408 USA (e-mail: zhenk@iu.edu).

Jongmo Sung, Mi Suk Lee, and Seungkwon Beack are with the Electronics and Telecommunications Research Institute, Daejeon 34129, Korea (e-mail: jmseong@etri.re.kr; lms@etri.re.kr; skbeack@etri.re.kr).

Minje Kim is with the Department of Intelligent Systems Engineering, Indiana University, Bloomington, IN 47408 USA (e-mail: minje@indiana.edu).

Digital Object Identifier 10.1109/TASLP.2021.3129353

and learn latent representations which may benefit other DNN-implemented downstream applications, such as speech enhancement [14], [15], speaker identification [16] and automatic speech recognition [17], [18]. Having that, a neural codec can serve as a trainable acoustic unit integrated in future digital signal processing engines [13].

Recently proposed neural speech codecs have achieved high coding gain and reasonable quality by employing deep autoregressive models. The superior speech synthesis performance achieved in WaveNet-based models [19] has successfully transferred to neural speech coding systems, such as in [20], where WaveNet serves as a decoder synthesizing wideband speech samples from a conventional non-trainable encoder at 2.4 kbps. Although its reconstruction quality is comparable to waveform codecs at higher bitrates, the computational cost is significant due to the model size of over 20 million parameters.

Meanwhile, VQ-VAE [12] integrates a trainable vector quantization scheme into the variational autoencoder (VAE) [21] for discrete speech representation learning. While the bitrate can be lowered by reducing the sampling rate 64 times, the downside for VQ-VAE is that the prosody can be significantly altered. Although [22] provides a scheme to pass the pitch and timing information to the decoder as a remedy, it does not generalize to non-speech signals. More importantly, VQ-VAE as a vocoder does not address the complexity issue since it uses WaveNet as the decoder. Although these neural speech synthesis systems noticeably improve the speech quality at low bitrates, they are not feasible for real-time speech coding on the hardware with limited memory and bandwidth.

LPCNet [23] focuses on efficient neural speech coding via a WaveRNN [24] decoder by leveraging the traditional linear predictive coding (LPC) techniques. The input of the LPCNet is formed by 20 parameters (18 Bark scaled cepstral coefficients and 2 additional parameters for the pitch information) for every 10 ms frame. All these parameters are extracted from the non-trainable encoder, and vector-quantized with a fixed codebook. As discussed previously, since LPCNet functions as a vocoder, the decoded speech quality is not considered transparent [1].

In this paper, we propose a novel neural speech coding system, with a lightweight design and scalable performance. First, we design a generic neural waveform codec with only 0.35 million parameters where 0.12 million parameters belong to the decoder. Compared to our previous models in [25], [26] where the decoder has 0.23 million parameters, the current neural codec employs gated linear units to boost the gradient flow during model training and depthwise separable convolution to achieve further efficiency during decoding, as detailed in Section II. Based on this neural codec, our full system features two mechanisms to integrate speech production theory and residual coding techniques in Section III. Benefited from the residual-excited linear prediction (RELP) [27], we conduct LPC and apply the neural waveform codec to the excitation signal, which is illustrated in Section III-A. In this integration, a trainable quantizer bridges the encoding of linear spectral pairs and the corresponding LPC residual, making the speech coding pipeline end-to-end trainable. We also enable residual coding among neural waveform codecs to scale up the performance for high bitrates (Section III-B). In summary, the proposed system has following characteristics:

- *Scalability:* Similar to LPCNet [23], the proposed system is compatible with conventional spectral envelope estimation techniques. However, ours operates at a much wider bitrate range with comparable or superior speech quality to standardized waveform codecs.
- *Compactness:* The neural waveform codec in our system is with a much lower complexity than WaveNet [19] and VQ-VAE [12] based codecs. Our decoder contains only 0.12 million parameters which is $160\times$ more compact than a WaveNet counterpart. Our TensorFlow implementation's execution time to encode and decode a signal is only 42.44% of its duration on a single-core CPU in the low-to-medium bitrates and 80.21% in the high bitrate, facilitating real-time communications.
- *Trainability:* Our method is with a trainable encoder as in VQ-VAE, which can be integrated into other DNNs for acoustic signal processing. Besides, it is not constrained to speech, and can be generalized to audio coding with minimal effort as shown in [28].

TABLE I
CATEGORICAL SUMMARY OF RECENTLY PROPOSED NEURAL SPEECH CODING SYSTEMS. ✔ MEANS THE SYSTEM SUPPORTS THE FEATURE WHILE ✗ DOES NOT. ● MEANS IT IS NOT KNOWN

	WaveNet [20]	VQ-VAE [22]	LPCNet [23]	Proposed
Transparent coding	✔	●	✗	✔
Less than 1M parameters	✗	✗	✔	✔
Real-time communications	✗	✗	✔	✔
Encoder trainable	✔	✔	✗	✔

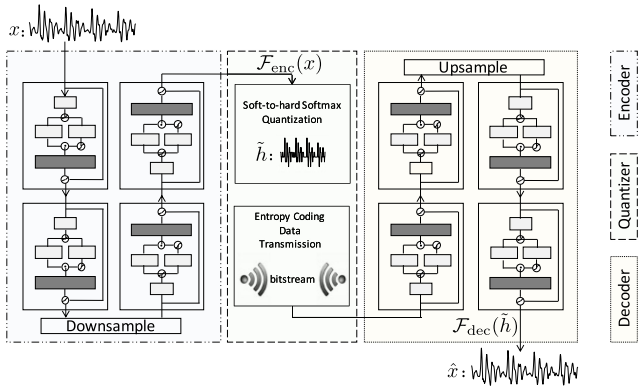
Table I highlights the comparison to the other existing neural speech codecs.

This paper is an extension of the authors' previous conference presentations [25], [26], where some initial ideas were already discussed. The new contributions presented this journal version are listed as follows:

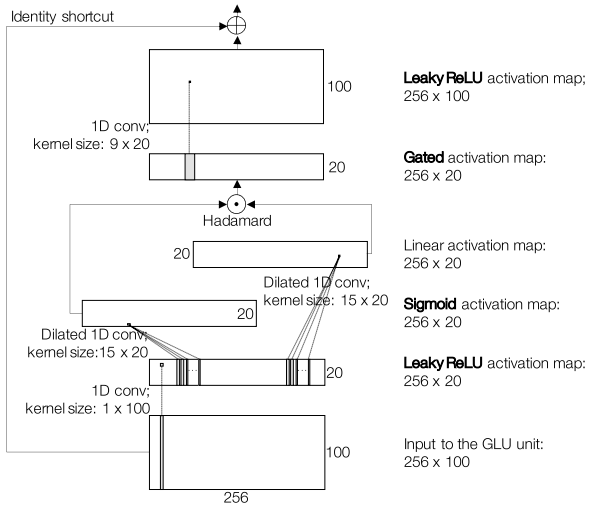
- *Novel Algorithmic Enhancements:* We propose new neural network architectures to form a new baseline autoencoder module and used it everywhere in our codecs. In our previous works, we have used a 1D convolutional neural network (CNN) that defines an autoencoder block with an identity shortcut as in the ResNet architecture [29]. While this architecture has been effective, in this journal paper, we propose to use the dilated gated linear units and depthwise separable convolution to reduce the kernel size without inducing any performance degradation. Consequently, our NWC is defined by 0.35 M parameters, whose decoder part accounts for only 0.12 M parameters. Compared to our previous models that are already small with only 0.45 M parameters, the newly introduced reduction amounts to 22.2%. If we only compare the decoder parts, it is a 47.8% reduction. Although the proposed architecture is more compact than our previous models or the WaveNet-based codecs, since neural codecs' complexity is much larger than the traditional speech codecs, the additional model

complexity reduction with no degradation of performance is promising. The architectural improvement are presented in Section II-A.

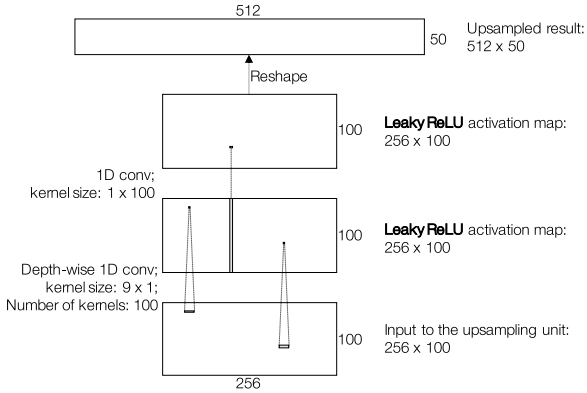
- *Extensive Experimental Validation:* In our previous works, the experimental validation was to prove the initial concepts individually proposed each paper. In this time, we conduct an extensive and thorough experiments to provide the readers with a full view to the whole building-blocks of the neural speech coding. To this end, we define four candidate systems, from Model-I to IV, by incrementally adding new modules, such as LPC, the trainable LPC quantizer, and multiple concatenated neural autoencoders. The objective and subjective tests validate each of these additions in a full view (Table III and Fig. 7).
- *Additional Analyses and Ablation Tests:* We also provide detailed experimental validation for most of the claims made in the paper by designing and performing separate experiments, which were missing in the previous papers.
  - Section IV-D1 provides experimental verification that the proposed compact neural architecture does not induce performance loss.
  - Section IV-D2 presents a detailed analysis of the behavior of the cascaded autoendoers and the impact of different training strategies.
  - Section IV-F1 explores contribution of different loss terms in our training objective by performing ablation tests, and then proposes an optimal combination of hyperparameters.
  - Section IV-F2 also conducts an ablation test to empirically verify that the proposed trainable LPC quantization algorithm improves speech quality at the same bitrate.
  - Section IV-F3 and IV-F4 analyze the bit allocation behavior among the different submodules. Since the bit allocation strategy is decided by the learning algorithm, these analyses provide evidence that our models dynamically adapt to the characteristics of the signals given the limited bit budget.
  - Section IV-G presents additional analyses on computational complexity and execution time ratios to discuss the potential of the neural codecs in real-time applications.
  - Last but not least, in Section IV-G3 we discuss the implementation issues and the limitations of the proposed system in the context of real-world application scenarios.



(a) The high-level structure of proposed neural waveform codec



(b) Dilated gated linear unit (GLU)



(c) Depthwise separable 1D convolution for upsampling

Fig. 1. The proposed architecture for lightweight NWC.

II. End-to-End Neural Waveform Codec (NWC)

The neural waveform codec (NWC), is an end-to-end autoencoder that forms the base of our proposed coding systems. NWC directly encodes the input waveform $\boldsymbol{x} \in \mathbb{R}^T$ using a convolutional neural network (CNN) encoder $\mathcal{F}_{\text{enc}}(\cdot)$, i.e., $\boldsymbol{h} \leftarrow \mathcal{F}_{\text{enc}}(\boldsymbol{x})$. Then, the quantization process $\mathcal{Q}(\cdot)$ converts the encoding into a bitstring $\tilde{\boldsymbol{h}} \in \mathbb{R}^N$, which is followed by lossless data compression and bitstream transmission. On the receiver side, the decoder reconstructs the waveform as $\boldsymbol{x} \approx \hat{\boldsymbol{x}} \leftarrow \mathcal{F}_{\text{dec}}(\tilde{\boldsymbol{h}})$. Fig. 1(a) depicts NWC's overall system architecture. The structure is detailed in Table II. It serves as a basic component in the proposed speech coding system in Section III. In this section, we first introduce the architectural improvement that reduced our model's complexity. Next, we also introduce two strategies

TABLE II
ARCHITECTURE OF THE NEURAL WAVEFORM CODEC: INPUT AND OUTPUT TENSORS ARE SHAPED AS (SAMPLE, CHANNEL), WHILE THE KERNEL IS REPRESENTED AS (KERNEL SIZE, IN CHANNEL, OUT CHANNEL)

	Layer	Input shape	Kernel shape	Output shape
Encoder	Channel Expansion	(512, 1)	(55, 1, 100)	(512, 100)
	Gated Linear Unit	(512, 100)	[(1, 100, 20), (15, 20, 20)†, (15, 20, 20)†, (9, 20, 100)] ×2	(512, 100)
	Downsampling	(512, 100)	(9, 100, 100)	(256, 100)
	Gated Linear Unit	(256, 100)	[(1, 100, 20), (15, 20, 20)†, (15, 20, 20)†, (9, 20, 100)] ×2	(256, 100)
	Channel Reduction	(256, 100)	(9, 100, 1)	(256, 1)
Decoder	Channel Expansion	(256, 1)	(9, 1, 100)	(256, 100)
	Gated Linear Unit	(256, 100)	[(1, 100, 20), (15, 20, 20)†, (15, 20, 20)†, (9, 20, 100)] ×2	(256, 100)
	Upsampling	(256, 100)	(9, 100, 1), (1, 100, 100)	(512, 50)
	Gated Linear Unit	(512, 50)	[(1, 50, 20), (15, 20, 20)†, (15, 20, 20)†, (9, 20, 50)] ×2	(512, 50)
	Channel Reduction	(512, 50)	(55, 50, 1)	(512, 1)

that compress the signals: feature map compression and trainable quantization.

A. The Improved Architecture for NWC

We propose two different kinds of structural modification to reduce the model's overall complexity compared to other NWC models including our prior works [25], [26].

First, both encoder and decoder adopt gated linear units (GLU) [30]. We also define the GLU's convolution with dilation [31] to expand the receptive field in the time domain, which is a scheme that showed promising performance in speech enhancement [32]. Fig. 1(b) shows our dilated GLU module. It first reduces the channel from 100 to 20 using a unit-width kernel. Then, two separate dilated convolution layers are applied to produce two feature maps, one of which goes through a sigmoid activation. Hence, the Hadamard product of the two feature maps can be seen as a *gated* version of the linear feature map. It is known that this gating mechanism in the middle boosts the gradient flow thanks to the linear path that does not involve the gradient vanishing issue. The final result is a mixture of the input and the last feature map, turning this block into a residual learning function as proposed in ResNet [29]. This kind of architecture shows superior performance as evidenced in [30], [33].

Our encoder reduces the data rate by using a "downsampling" convolutional layer, whose "stride" parameter is set to be 2. As a counterpart, the decoder's "upsampling" layer makes up the loss. More details about this down and upsampling operations will be discussed in Section II-B. In this subsection, we focus on the actual module that performs the upmixing, where we introduced additional reduction in complexity. Out of various choices, we employ the depthwise separable convolution [34] to further save the computational cost (Fig. 1(c)). For example, to transform a feature map of size 256×100 (features, channels) into its upsampled version of size 512×50, we first perform depthwise convolution using a $c \times 1$ kernel. In our system $c = 9$. Since the depthwise convolution applies to each channel separately, we eventually need 100 such kernels. It is a reduction of model complexity, because a normal convolution requires a kernel of size $c \times 100 \times 100$ (features, input channels, output channels), which is 100 time larger. In depthwise separable convolution, another 1×1 convolution follows to add more nonlinearity, for which we need a $1 \times 100 \times 100$ kernel. In our case, it is easy to show that $c \times 100 \times 100 > c \times 100 + 100 \times 100$ when the integer $c > 1$. For example, when $c = 9$, it is a reduction of about 88% of parameters.

Likewise, the proposed NWC is lightweight with only 0.35 million parameters, which is a reduction of 0.1 million parameters compared to our previous works [25], [26]. The reduction comes from the streamlined upsampling operation implemented via the depthwise separable convolution. Eventually, the decoder accounts for 0.12 million parameters out of 0.35.

B. Feature Map Compression

One way to compress the input signal in the proposed encoder architecture is to reduce the data rate. The CNN encoder function takes an input frame, $\boldsymbol{x} \in \mathbb{R}^T$, and converts it into a feature map $\boldsymbol{h} \in \mathbb{R}^N$,

$$\boldsymbol{h} \leftarrow \mathcal{F}_{\text{enc}}(\boldsymbol{x}), \tag{1}$$

which then goes through quantization, transmission, and decoding to recover the input as shown in Fig. 1(a). During the encoding process, we introduce a *downsampling* operation, reducing the dimension of the code vector $\boldsymbol{h}$. We employ a dedicated downsampling layer by setting up the stride value to be 2 during its convolution, reducing the data rate by 50%, i.e., $N = T/2$. Accordingly, the decoder needs a corresponding upsampling operation to recover the original sampling rate. We use subpixel CNN layer proposed in [35] to recover the original sampling rate. Concretely, the subpixel upsampling involves a feature transformation implemented in depthwise convolution, and a shuffle operation that interlaces features from two channels into a single channel, as shown in Eq. (2), where the input feature of the shuffle operation is shaped as $(N, 2)$ and the output is shaped as $(2N, 1)$.

$$\begin{aligned}&[h_{11}, h_{21}, h_{12}, h_{22}, \ldots, h_{1N}, h_{2N}]\\ &\leftarrow \text{Upsampling}([h_{11}, h_{12}, \ldots, h_{1N}; h_{21}, h_{22}, \ldots, h_{2N}] \end{aligned} \tag{2}$$

C. The Trainable Quantizer for Bit Depth Reduction

The dimension-reduced feature map can be further compressed via bit depth reduction. Hence, the floating-point code $\boldsymbol{h}$ goes through quantization and entropy coding, which will finalize the bitrate based on the entropy of the code value distribution. Typically, a bit depth reduction procedure lowers the average amount of bits to represent each sample. In our case, we could employ a quantization process that assigns the output of the encoder to one of the pre-defined quantization bins. If there are $2^5 = 32$ quantization bins, for example, a single-precision floating-point value's bit depth reduces from 32 to 5. In addition, various entropy coding techniques, such as Huffman coding, can be further employed to losslessly reduce the bit depth. While the quantization could be done in a traditional way, e.g., using Lloyds-Max quantization [36] *after* the neural codec is fully trained, we encompass the quantization step as a trainable part of the neural network as proposed in [37]. Consequently, we expect that the codec is aware of the quantization error, which the training procedure tries to reduce it. It is also convenient to control the bitrate by controlling the entropy of the code value distribution, which can be also done as a part of network training.

In NWC, the quantization process is represented as classification on each scalar value of the encoder output. Given a vector with K centroids, $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_K]^\top$, the quantizer's goal is to assign each feature h_n to the closest centroid in terms of ℓ_2 distance, which is defined as follows:

$$\boldsymbol{D} = \begin{bmatrix} ||h_1 - \beta_1||_2 & \cdots & ||h_1 - \beta_K||_2 \\ \vdots & \ddots & \vdots \\ ||h_N - \beta_1||_2 & \cdots & ||h_N - \beta_K||_2 \end{bmatrix}, \tag{3}$$

where n-th row in $\boldsymbol{D}$ is a vector of ℓ_2 distance between n-th code value h_n to all K quantization bins. Then, we employ the softmax function to turn each row of $\boldsymbol{D}$ into a K-dimensional probabilistic assignment vector:

$$\boldsymbol{A}^{(\text{soft})} = \begin{bmatrix} \text{softmax}(-\alpha \boldsymbol{D}_{1:}) \\ \text{softmax}(-\alpha \boldsymbol{D}_{2:}) \\ \vdots \\ \text{softmax}(-\alpha \boldsymbol{D}_{N:}) \end{bmatrix}, \tag{4}$$

where we turn the distance into a similarity metric by multiplying a negative number $-\alpha$, such that the shortest distance is converted to the largest probability. In our implementation, $\boldsymbol{\beta}$ is initialized as a vector of $K = 32$ uniformly spaced numbers within the interval of $[-1, 1]$. As for α, we begin with a large enough number 300. Both α and β are trainable parameters to optimize the quantization process.

Note that (4) yields a soft assignment matrix $\boldsymbol{A}^{(\text{soft})} \in \mathbb{R}^{N \times K}$. In practice, though, the quantization process must perform a hard assignment, so each code value h_n is replaced by an integer index to the closest centroids: $z_n \in \{1, 2, \ldots, K\}$, which is represented by $\lceil \log_2 K \rceil$ bits as the quantization result. The hard kernel assignment matrix $\boldsymbol{A}^{(\text{hard})}$, where each row is a one-hot vector, can be induced by turning on the maximum element of $\boldsymbol{A}^{(\text{soft})}$ while suppressing the non-maximum:

$$\boldsymbol{A}^{(\text{hard})}_{nk} = \begin{cases} 1 \ \text{ if arg max}_{\ j \in \{1,2,\ldots,K\}} \boldsymbol{A}^{(\text{soft})}_{nj} = k \\ 0 \ \text{otherwise} \end{cases} \tag{5}$$

On the decoder side, $\tilde{\boldsymbol{h}} = \boldsymbol{A}^{(\text{hard})}\boldsymbol{\beta}$ recovers $\boldsymbol{h}$.

Since $\arg\max$ operation in (5) is not differentiable, a soft-to-hard scheme is proposed in [37], where $\boldsymbol{A}^{(\text{hard})}$ is used only at test time. During backpropagation for training, the soft classification mode is enabled with $\boldsymbol{A}^{(\text{soft})}$ so as not to block the gradient flow. In other words, $\hat{\boldsymbol{h}} = \boldsymbol{A}^{(\text{soft})}\boldsymbol{\beta}$ represents each encoder output with a linear combination of all quantization bins. The process is summarized in Algorithm 1. Although this soft quantization process is differentiable and desirable during training, the discrepancy between $\boldsymbol{A}^{(\text{soft})}$ and $\boldsymbol{A}^{(\text{hard})}$ creates higher error during the test time, requiring a mechanism to reduce the discrepancy as in the following section.

Algorithm 1: Soft-to-Hard Quantization During Inference, $\mathcal{Q}(\boldsymbol{h}, \alpha, \boldsymbol{\beta})$.

Input: the code, e.g., the encoder output, $\boldsymbol{h} = \mathcal{F}_{\text{enc}}(\boldsymbol{x})$
the trained softmax scaling factor, α
the trained centroid vector, $\boldsymbol{\beta} \in \mathbb{R}^K$
Output: the quantized code, $\hat{\boldsymbol{h}}$ (training) or $\tilde{\boldsymbol{h}}$ (testing)
Compute the dissimilarity matrix: $\boldsymbol{D}_{nk} \leftarrow \ell_2(h_n || \beta_k)$
Softmax conversion: $\boldsymbol{A}^{(\text{soft})}_{n:} \leftarrow \text{Softmax}(-\alpha \boldsymbol{D}_{n:})$
if Training **then**
    Soft quantization: $\hat{\boldsymbol{h}} \leftarrow \boldsymbol{A}^{(\text{soft})}\boldsymbol{\beta}$
else if Testing**then**
    Hard quantization: $\tilde{\boldsymbol{h}} \leftarrow \boldsymbol{A}^{(\text{hard})}\boldsymbol{\beta}$
end if

1) Soft-to-Hard Quantization Penalty: Although the limit of $\boldsymbol{A}^{(\text{soft})}$ is $\boldsymbol{A}^{(\text{hard})}$ as α approaches ∞, the change of α should be gradual to allow gradient flows in the initial phase of training. We control the *hardness* of $\boldsymbol{A}^{(\text{soft})}$ using the soft-to-hard quantization loss derived from [38]:

$$\mathcal{L}_{\text{Q}} = \frac{1}{N} \sum_{n,k} \sqrt{\boldsymbol{A}^{(\text{soft})}_{nk}}, \tag{6}$$

whose minimum, 1, is achieved when $\boldsymbol{A}^{(\text{soft})}_{n:}$ is a one-hot vector for all n. Conversely, when $\boldsymbol{A}^{(\text{soft})}_{nk} = 1/K$, the loss is maximum. Hence, by minimizing this soft-to-hard quantization penalty term, we can regularize the model to have *harder* $\boldsymbol{A}^{(\text{soft})}$ values by updating α and the other model parameters accordingly. As a result, the test time quantization loss will be reasonably small when $\boldsymbol{A}^{(\text{soft})}$ is replaced by $\boldsymbol{A}^{(\text{hard})}$.

2) Bitrate Calculation and Entropy Control: The bitrate is calculated as a product of the number of code values per second and the average bit depth for each code. The former is defined by the dimension of the code vector N multiplied by the number of frames per second, $\frac{F}{T-o}$, where T, o, and F are the input frame size, overlap size, and the original sampling rate, respectively. If we denote the average bit depths per sample by a function $g(\tilde{h}_n)$, the bitrate can be computed as in (7),

$$\text{bitrate} = g(\tilde{h}_n) NF/(T - o). \tag{7}$$

When $F = 16,000, T = 512, o = 32$, and $N = 256$ after downsampling, for example, there are about 8,533 samples per second. If $g(\tilde{h}_n) = 3$ bits, the bitrate is estimated as 25.6 kbps. In contrast, the uncompressed bitrate is 256 kbps with $N = T = 512$, $o = 0$, and $g(x_t) = 16$ bits for each sample.

We alter the entropy of $\boldsymbol{\beta}$ to adjust the codec's bitrate, since the entropy serves as the lower bound of $g(\tilde{h}_n)$ based on Shannon's entropy theory. The entropy, $\mathcal{H}(\boldsymbol{\beta})$, is estimated with the sample distribution,

$$\mathcal{H}(\boldsymbol{\beta}) \approx -\sum_{k=1}^{K} p(\beta_k) \log_2 p(\beta_k), \tag{8}$$

where $p(\beta_k) = \frac{1}{N} \sum_n \boldsymbol{A}_{nk}^{(\text{hard})}$ is the relative frequency of the k-th centroid being chosen during quantization. To navigate the model training towards the target bitrate, $\mathcal{H}(\boldsymbol{\beta})$ defined in (8) is included in the loss function as a regularizer: if the current bitrate is higher than desired, the optimization process will increase the blending weight of the regularizer to strengthen the regularization effect, which consequently lowers the entropy, and vice versa. More details on the training target and hyperparameter setting are discussed in Section IV-B.

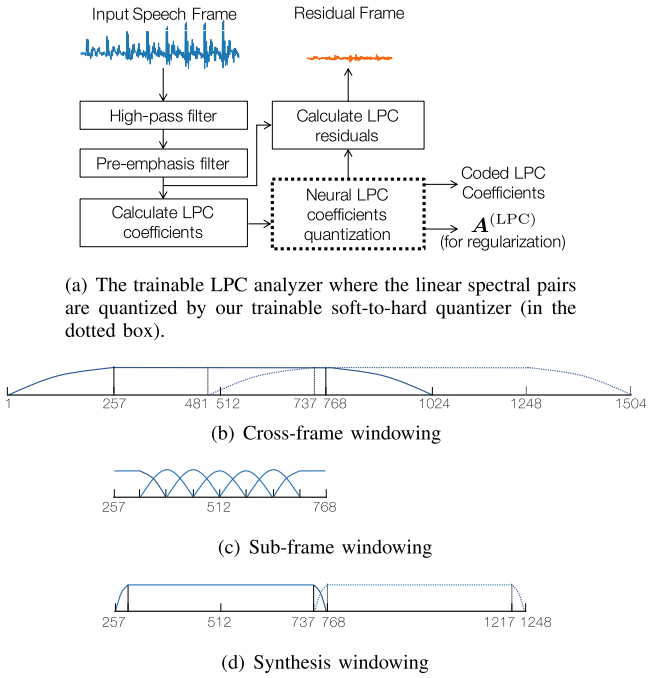
III. NWC-Based Speech Coding Systems

By having NWC introduced in Section II as the basic module, we propose two different extension mechanisms to improve the codec's performance in a wider range of bitrates, without increasing the model complexity significantly. First, in Section III-A, we propose a neural network compatible LPC module where the trainable soft-to-hard quantization is applied to the LPC coefficients. With the LPC module followed by an NWC module, we achieve a win-win strategy that fuses the traditional DSP technique and the modern deep learning model [26]. In addition to integrating LPC, our neural codec conducts multistage residual coding [25] by cascading residuals among multiple NWC modules (Section III-B). The proposed CMRL system relays residual signals among the series of NWCs to scale up the coding performance at high bitrates.

A. Trainable LPC Analyzer

LPC has been widely used to facilitate speech compression and sysnthesis, where *source-filter* model "explains out" the envelope of a speech spectrum, leaving a low-entropy residual signal [39]. Similarly, LPC serves as a pre-processor in our system before its residual signal being compressed by NWC as we will see in Section III-B. In this subsection, we redesign the LPC coefficient quantization process as a trainable module. We introduce collaborative quantization (CQ) to jointly optimize the LPC analyzer and NWCs as a residual coder.

1) Speech Resonance Modeling: In the speech production process, the source as wide-band excitation signals go through the vocal tract tube. The shape-dependent resonances of the vocal tract filter the excitations before it being transformed to speech signals [40]. In speech coding, the "vocal tract response" is often modeled as an all-pole filter [41]. Having that, the t-th sample x_t can be approximated by a autoregressive model using



(a) The trainable LPC analyzer where the linear spectral pairs are quantized by our trainable soft-to-hard quantizer (in the dotted box).

(b) Cross-frame windowing

(c) Sub-frame windowing

(d) Synthesis windowing

Fig. 2. The signal flow chart for LPC analyzer (a) and windowing schemes in LPC (b)-(d).

M previous samples,

$$x_t = \sum_{k=1}^{M} l_k x_{t-k} + e_t, \tag{9}$$

where the estimation error e_t represents the LPC residual, and l_k denotes the filter coefficients. Typically, l_k can be efficiently estimated via Levinson-Durbin algorithm [42], and are to be quantized before LPC residual is calculated, i.e., e_t encompasses the quantization error. The LPC residual e_t serves as input to the NWC module, which works as explained in Section II, but on $\boldsymbol{e}$ rather than $\boldsymbol{x}$. Hence, how LPC coefficients are quantized determines NWC's input, the LPC residual.

2) Collaborative Quantization: The conventional LPC coefficient quantization process is standardized in ITU-T G.722.2 (AMR-WB) [43]: 2.4 k bits are assigned to represent the LPC coefficient per second though multistage vector quantization (MSVQ) [44] in a classic LPC analyzer. Once again, we employ the soft-to-hard quantizer as illustrated in Section II to make the quantization and bit allocation steps in the LPC analyzer trainable and communicatable with the neural codec.

We compute the LPC coefficients as in [5], first by applying high-pass filtering followed by pre-emphasizing (Fig. 2(a)). When calculating LPC coefficients, the window in Fig. 2(b) is used. The window is symmetric with the left and right 25% parts being tapered by a 512-point Hann window. After representing the 16 LPC coefficients in linear spectral pairs (LSP) [45], we quantize it using the soft-to-hard quantization scheme. Then, the sub-frame window in Fig. 2(c) is applied to calculate LPC residual, which assures a more accurate residual calculation. The

frame that covers samples [256:768], for instance, is decomposed into 7 sub-frames to calculate LPC residuals separately. Each 128-point Hann window in Fig. 2(c) is with 50% overlap, except for the first and last window. They altogether form a constant overlap-add operation. Finally, after the synthesis using the reconstructed residual signal and corresponding LPC coefficients, the window in Fig. 2(d) tapers both ends of the synthesized signal, covering 512 samples with 32 overlapping samples between adjacent windows.

As an intuitive example, given the samples [1:1024] as the input, after the LPC analysis, neural residual coding, and LPC synthesis, samples [257:768] are decoded; the next input frame is [481:1504] (the dotted window in Fig. 2(b)), whose decoded samples are within [737:1248]. The overlap-add operation is applied to the final decoded samples [737:768] (Fig. 2(d)).

During this process, the calculated LPC coefficients are quantized using Algorithm 1, where the code vector is with 16 dimensions, i.e., $\boldsymbol{h} \in \mathbb{R}^{16}$. The number of kernels is set to be $K = 2^8 = 256$. Note that the soft assignment matrix for the LPC quantization, $\boldsymbol{A}^{(\mathrm{LPC})}$, is also involved in the loss function to regularize the bitrate.

We investigate the impact of the trainable LPC quantization in collaboration with the rest of the NWC modules in Section IV.

B. Cross-Module Residual Learning (CMRL)

To achieve scalable coding performance towards transparency at high bitrates, we propose cross-module residual learning (CMRL) to conduct bit allocation among multiple neural codecs in a cascaded manner. CMRL can be regarded as a natural extension of what is described in Section III-A, where the LPC as a codec conducts the first round of coding by only modeling the spectral envelope. It leaves the residual signal for a subsequent NWC to be further compressed. With CMRL, we employ the concept of residual coding to cascade more NWCs. We also present a dual-phase training scheme to effectively train the CMRL model.

CMRL's scalability comes from its residual coding concept that enables a concatenation of multiple autoencoding modules. We define the residual signal recursively: i-th codec takes the residual of its predecessor as input, and the i-th reconstruction creates another residual for the next round, and so on. Hence, we have

$$\hat{\boldsymbol{x}}^{(i)} \leftarrow \mathcal{F}^{(i)}(\boldsymbol{x}^{(i)}), \boldsymbol{x}^{(i)} \leftarrow \boldsymbol{x}^{(i-1)} - \hat{\boldsymbol{x}}^{(i-1)}, \boldsymbol{x}^{(1)} \leftarrow \boldsymbol{x}, \quad (10)$$

where $\hat{\boldsymbol{x}}^{(i)}$ stands for the reconstruction of the i-th input using the i-th coding module $\mathcal{F}^{(i)}(\cdot)$, while the input to the first codec is defined by the raw input frame $\boldsymbol{x}$. If we expand the recursion, we arrive at the non-recursive definition of $\boldsymbol{x}^{(i)}$,

$$\boldsymbol{x}^{(i)} = \boldsymbol{x} - \sum_{j=1}^{i-1} \hat{\boldsymbol{x}}^{(j)}, \quad (11)$$

which means the input to i-th model is the residual of the sum of all preceding $i-1$ codecs' decoded signals. It ensures the additivity of the entire system: adding more modules keeps improving the reconstruction quality. Hence, CMRL can scale up to high bitrates at the cost of increased model complexity.

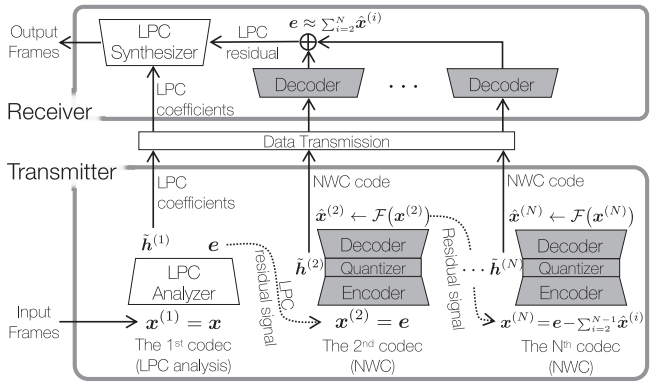


Fig. 3. The flow diagram of the test-time inference.

CMRL is optimized in two phases. During Phase-I training, we sequentially train each codec from the first to the last one using a module-specific residual reconstruction goal,

$$\mathcal{E}(\boldsymbol{x}^{(i)} || \hat{\boldsymbol{x}}^{(i)}). \quad (12)$$

The purpose for Phase-I training is to get parameters for each codec properly initialized. Then, Phase-II finetunes all trainable parameters of the concatenated modules to minimize the global reconstruction loss,

$$\mathcal{E}\left(\boldsymbol{x} \middle|\middle| \sum_{i=1}^{N} \hat{\boldsymbol{x}}^{(i)}\right). \quad (13)$$

The reconstruction loss measures the waveform discrepancy in both time and frequency domains. Quantization penalty and entropy control are introduced as regularizers. Section IV-B details the definition of the training target and ablation tests on how to find the optimal blending weights between the loss terms.

C. Signal Flow During Inference

Fig. 3 shows the full system signal flow with N sub-codecs, having an LPC module as the first one. On the transmitter side the LPC analyzer first processes the input frame $\boldsymbol{x}$ of 1024 samples and computes 16 coefficients, $\boldsymbol{h}^{(1)}$, as well as 512 residual samples $\boldsymbol{x}^{(2)}$ at the center of the frame. Then, the residual signal goes through the $N-1$ NWCs in sequentially. Note that the transmission process's primary job is to produce a quantized bitstring $\tilde{\boldsymbol{h}}^{(i)}$ from LPC and each NWC. To this end, NWC's decoder part must also run to compute the residual signal and relay it to the next NWC module. The bitstring is generated as a concatenation of all encoder outputs: $\tilde{\boldsymbol{h}} = [\tilde{\boldsymbol{h}}^{(1)}; \tilde{\boldsymbol{h}}^{(2)}; \cdots; \tilde{\boldsymbol{h}}^{(N)}]$. Once the bitstring is available on the receiver side, all NWC decoders run to reconstruct the LPC residual signal, i.e., $\hat{\boldsymbol{x}}^{(2)} \approx \sum_{i=2}^{N} \mathcal{F}_{\mathrm{dec}}^{(i)}(\tilde{\boldsymbol{h}}^{(i)})$. Then it is used as input of the LPC synthesizer, along with the LPC coefficients.

IV. Evaluation

In this section, we examine the proposed neural speech coding model presented in Section II and III. The evaluation critera include both objective measures such as PESQ [46] and signal-to-noise ratio (SNR) and subjective scores from MUSHRA listening tests [47]. In addition, we conduct ablation analysis to provide a detailed comparison between various loss terms and bit allocation schemes. Finally, we report the system delay and execution time under four hardware specifications.

A. Data Processing

The training dataset is created from 300 speakers randomly selected from the TIMIT corpus [48] with no gender preference. Each speaker contributes 10 utterances totaling 2.6 h-long training set, which is a reasonable size due to our compact design. The same scheme is adopted when creating the validation dataset and test dataset with 50 speakers, respectively. All three datasets are mutually exclusive with the sample rate of 16 kHz. All neural codecs in this work are trained and tested via the same set of data for a fair comparison. We normalize each utterance to have a unit variance, then divided by the global maximum amplitude, before being framed into segments with the size of 512 samples. On the receiver side, we conduct overlap-and-add after the synthesis of the frames, where a 32-sample Hann window is applied to the overlapping region of the same size.

With the LPC codec, we apply high-pass filtering defined in the z-space, $\mathcal{G}_{\text{hp}}(z) = \frac{0.989502 - 1.979004 z^{-1} 0.989502 z^{-2}}{1 - 1.978882^{-1} 0.979126^{-2}}$, to the normalized waveform. A pre-emphasis filter, $\mathcal{G}_{\text{preemp}}(z) = 1 - 0.68 z^{-1}$, follows to boost the high frequencies.

B. Training Targets and Hyperparameters

The loss function is defined as

$$\mathcal{L} = \lambda_{\text{MSE}} \sum_{t=1}^{T} (x_t - \hat{x}_t)^2 + \lambda_{\text{mel}} \sum_{b=1}^{4} \sum_{f=1}^{F_b} \left(y_f^{(b)} - \hat{y}_f^{(b)} \right)^2 + \lambda_{\text{Q}} \mathcal{L}_{\text{Q}} + \lambda_{\text{ent}} \mathcal{H}(\boldsymbol{\beta}) \tag{14}$$

where the first term measures the mean squared error (MSE) between the raw waveform samples and their reconstruction. Ideally, if the model complexity and the bitrate is sufficiently large, an accurate reconstruction is feasible by using MSE as the only loss function. Otherwise, the result is usually sub-optimal due to the lack of bits: coupled with the MSE loss, the decoded signals tend to contain broadband artifact. The second term supplements the MSE loss and helps suppress this kind of artifact. To this end, we follow the common steps to conduct mel-scaled filter bank analysis, which results in a mel spectrum $\boldsymbol{y}$ that has a higher resolution in the low frequencies than in the high frequencies. The filter bank size defines the granularity level of the comparison. Following [38], we conduct a coarse-to-fine filter bank analysis by setting four filter bank sizes, $F_1 = 8, F_2 = 16, F_3 = 32, F_4 = 128$ as shown in Fig. 4, which result in four kinds of resolutions for mel spectra $\boldsymbol{y}^{(b)}$ indexed by $b \in \{1, 2, 3, 4\}$.

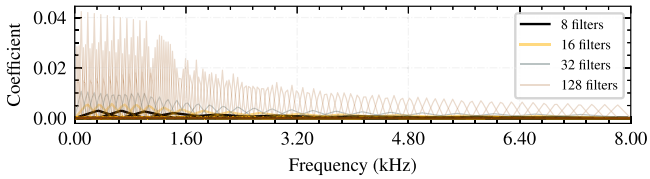


Fig. 4. The coarse-to-fine filter bank analysis in the mel scale.

All models are trained on Adam optimizer with default learning rate adaptation rates [49]. The batch size is fixed with 128 frames. The initial learning rate is 2×10^{-3} for the first neural codec. With CMRL, the learning rate for the successive neural codecs is 2×10^{-4}. Finetuning of all those models is with a smaller learning rate 2×10^{-5}. All models are sufficiently trained until the validation loss converges after being exposed to about 5×10^5 batches. These hyperparameters were chosen based on validation.

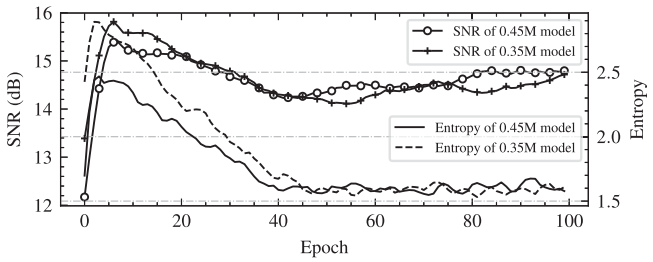
The blending weights in the loss function in (14) are also selected based on the validation performance. Empirically, the ratio between the time-domain loss and mel-scaled frequency loss affects the trade-off between the SNR and perceptual quality of decoded signals. If the time-domain loss dominates the optimization process, the model compresses each sub-band with an equal effort. In that case, the artifact will be audible unless the SNR reaches a rather high level (over 30 dB) which entails a high bitrate and model complexity. On the other hand, if only the mel-scaled frequency loss is in place, the reconstruction quality in the high frequency will degrade. The impact of these blending weights for these two loss terms is detailed in Section IV-F via an ablation analysis.

The weights for the quantization regularizer λ_{Q} and entropy regularizer λ_{ent} are initially set to be 0.5 and 0.0, respectively. As for λ_{ent}, we alter it after every epoch by 0.015: if the current model's bitrate is higher than the target bitrate, λ_{ent} increases to penalize the model's entropy more; otherwise, λ_{ent} decreases to boost the entropy. Note that we omit the module index i in (14), so the meaning of $\hat{x}_t$ depends on the context: either the module-specific reconstruction as in (12) or the sum of all recovered residual signals for Phase-II finetuning as in (13). Similarly, $\mathcal{L}_{\text{Q}}$ and $\mathcal{H}_{\boldsymbol{\beta}}$ can encompass all modules' quantization and entropy losses including LPC's for Phase-II. We delay the introduction of the quantization and entropy loss until the fifth epoch.

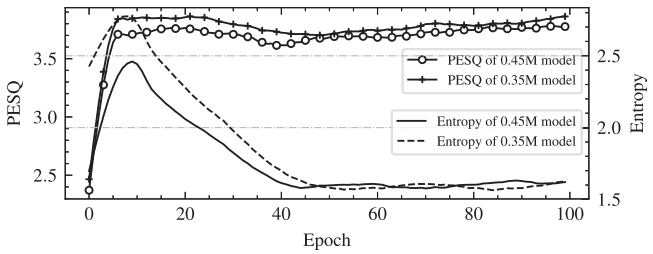
C. Bitrate Modes and Competing Models

We consider three bitrates, 12, 20, and 32 kbps, to validate models' performance in a range of use cases. We evaluate following different versions of neural speech coding systems:

- Model-I: The NWC baseline (Section II).
- Model-II: Another baseline that combines the legacy LPC and an NWC module for residual coding.
- Model-III: A trainable LPC quantization module followed by an NWC and finetuning (Section III-A);
- Model-IV: Similar to Model-III but with two NWC modules: the full-capacity CMRL implementation (Section III-B). It is tested cover the high bitrate case, 32 kbps.

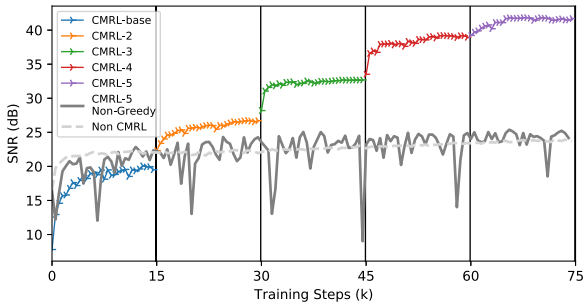


(a) The validation SNR curve during training

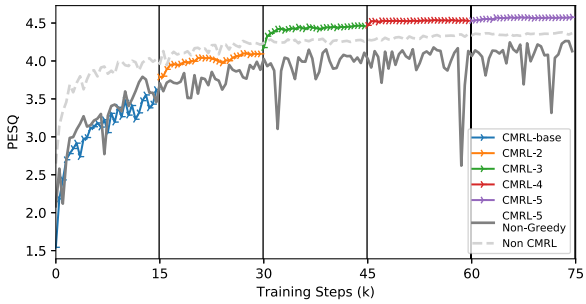


(b) The validation PESQ curve during training

Fig. 5. Speech reconstruction performance stays almost the same when the model size decreases from 0.45 to 0.35 million parameters with the help from the structural modification.



(a) Scalability with respect to SNR



(b) Scalability with respect to PESQ

Fig. 6. In CMRL, performance leaps when the new neural codec is added for residual cascading. In these CMRL models LPC is not used.

Regarding the standard codecs, AMR-WB [5] and Opus [50] are considered for comparison. AMR-WB, as an ITU standard speech codec, operates in nine different modes covering a bitrate range from 6.6 kbps to 23.85 kbps, providing excellent speech quality with a bitrate as low as 12.65 kbps in wideband mode. As a more recent codec, Opus shows the state-of-the-art performance in most bitrates up to 510 kbps for stereo audio coding, except for the very low bitrate range.

We first compare all models with respect to the objective measures, while being aware that they are not consistent with the subjective quality. Hence, we also evaluate these codecs in two rounds of MUSHRA subjective listening tests: the neural codecs are compared in the first round, whose winner is compared with other standard codecs in the second round.

D. Objective Measurements

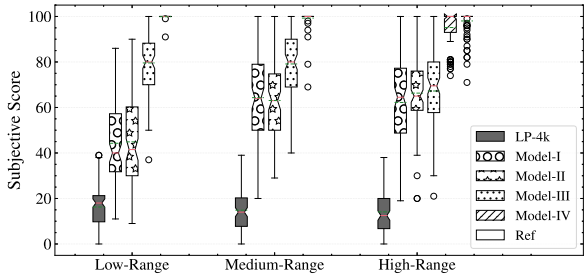
1) The Compact NWC Module and its Performance: Compared to our previous models in [25], [26] that use 0.45 million parameters, the newly proposed NWC in this work only has 0.35 million parameters. It is also a significant reduction from the other compact neural waveform codec [38] with 1.6 million parameters. As introduced in Section II the model size reduction is achieved via the GLU [33] and depthwise separable convolution for upsampling [34]. In our first experiment, we show that the objective measures stay the same. Fig. 5 compares the NWC modules before and after the structural modification proposed in Section II in terms of (a) signal-to-noise ratio (SNR) and (b) PESQ-WB [46]. We can see that the newly proposed model with 0.35 M parameters is comparable to the larger model. Therefore, it justifies its use as the basic module in a range of models from Model-I to IV.

2) The Impact of CMRL's Residual Coding: To validate the merit of CMRL's residual coding concept, we scale up the CMRL model by incrementally adding more NWC modules up to five. In Fig. 6, both SNR and PESQ values keep increasing when CMRL keeps adding a new NWC module. There are two noticeable points in these graphs. First, the greedy module-wise pretraining is important for the performance: whenever a new model is added, it is pretrained to minimize the module specific loss (12) first (Phase-I), then the global loss (13), subsequently (Phase-II). A model that does not perform Phase-II (thick gray line) stagnates no matter how many NWCs are added. Second, we also train a very large NWC model with the same amount of parameters as CMRL with five NWCs combined (grey dash). It turns out the equally large model fails to scale up due to its single integrated architecture. While we eventually decide to use only up to two NWCs for speech coding for our highest bitrate case, 32 kbps, we may keep adding NWCs to CMRL to meet the case of higher bitrates for non-speech audio coding.

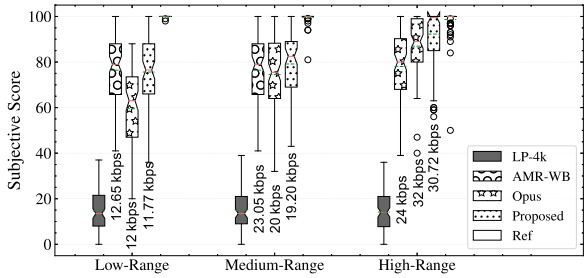
3) Overall Objective Comparison of All Competing Models: Table III reports SNR and PESQ-WB from all competing systems. AMR-WB in the low-range bitrate setting operates at 12.65 kbps and 23.05 kbps for the mid-range. Among neural speech coding systems, Model-I, as a single autoencoder model, outperforms others in all three bitrate setups in terms of SNR and PESQ-WB. It is also comparable to AMR-WB and Opus, except for the low bitrate case where Opus achieves the highest PESQ score. One explanation is that Model-I is highly optimized for the objective loss during training, although it does not necessarily mean that the higher objective score leads to a better subjective

TABLE III
OBJECTIVE MEASUREMENTS FOR NEURAL CODEC COMPARISON UNDER THREE BITRATE CASES

Bitrate (kbps)	SNR (dB)						PESQ-WB					
	Model-I	Model-II	Model-III	Model-IV	AMR-WB	Opus	Model-I	Model-II	Model-III	Model-IV	AMR-WB	Opus
∼12	**12.37**	10.69	10.85	–	11.60	9.63	3.67	3.45	3.60	–	3.92	**3.93**
∼20	**16.87**	10.73	13.65	–	13.14	9.46	**4.37**	3.95	4.01	–	4.18	**4.37**
∼32	**20.24**	11.84	14.46	17.11	–	17.66	**4.42**	4.15	4.18	4.35	–	4.38



(a) Neural waveform codecs comparison.



(b) Comparison between proposed methods and standard codecs.

Fig. 7. MUSHRA subjective listening test results.

quality as presented in Section IV-E. It is also observed that with CQ, Model-III gains slightly higher SNR and PESQ scores compared to Model-II, which uses the legacy LPC. Finally, the performance scales up significantly when Model-IV starts to employ two NWCs on top of LPC, which is our proposed full neural speech coding setup. Aside from objective measure comparison, to further evaluate the quality of proposed codec, we discuss the subjective test in the next section.

E. Subjective Test

We conduct two rounds of MUSHRA tests: (a) to select the best one out of the proposed models (from Model-I to IV) (b) to compare it with the standard codecs, i.e., AMR-WB and Opus. Each round covers three different bitrate ranges, totaling six MUSHRA sessions. A session consists of ten trials, for which ten gender-balanced test signals are randomly selected. Each trial has one low-pass filtered signal serving as the anchor (with a cutoff frequency at 4 kHz), the hidden reference, as well as signals decoded from competing systems. We recruit ten participants who are audio experts with prior experiences in speech/audio quality evaluation. The subjective scores are rendered in Fig. 7 as boxplots. Each box ranges from the 25 to 75 percentile with a 95% confidence interval. The mean and median are presented as the green dotted line and pink hard line, respectively. Outliers are represented in circles.

1) Comparison Among the Proposed Neural Codecs: In Fig. 7(a) we see that Model-III's produces decoding results that are much more preferred than both Model-I and Model-II, which are a pure end-to-end model and with the non-trainable legacy LPC module, respectively. The advantage is more noticeable in lower bitrates. It is contradictory to the objective scores reported in Table III where Model-I often achieves the highest scores. Compared to the deterministic quantization component in the legacy LPC in Model-II, LPC module and NWC in Model-III are jointly trained for a frame-wise independent bit allocation, so as to maximize the coding efficiency. We also note that Model-III's performance stagnates in the high bitrate experiments, suggesting its poor scalability. To this end, for the high bitrate experiment, we additionally test Model-IV with two NWC residual coding modules instead of just one. Model-IV outperformed both Model-II and Model-III by a large margin, showcasing a near-transparent quality.

2) Comparison With Standardized Codecs: Fig. 7(b) shows that, our Model-II is on par with AMR-WB for the low-range bitrate case, while outperforming Opus which tends to lose high frequency components. In the medium-range, Model-II at 19.2 kbps is comparable to Opus at 20.0 kbps and AMR-WB at 23.05 kbps. In the high bitrate range, our Model-IV outperforms Opus that operates 32 and 24 kbps, while AMR-WB is omitted as it does not support those high bitrates.

All MUSHRA sessions are available online, along with demo samples and source codes.[1]

F. Ablation Analysis

In this section, we perform some ablation analyses to justify our choices that led to CQ and CMRL's superior subjective test results. We investigate how different blending ratios between loss terms can alter the performance. We will also explore the optimal bit allocation strategy among coding modules.

1) Blending Weights for the Loss Terms: Out of the two reconstruction loss terms, MSE serves as the main loss for the end-to-end NWC system, while the mel-scaled loss prioritizes certain frequency bands over the others. We perform ablation analysis on four blending ratio settings to analyze their effect on decoded speech's objective quality. We consider both system configurations: one with only the neural waveform codec (Table IV(a)) and the other one with both the neural waveform codec and collaboratively trained LPC module (Table IV (b). For Table IV (a), the target entropy for each sample in the neural codec is of 2.5-bit, corresponding to a bitrate of

[1][Online]. Available: https://saige.sice.indiana.edu/research-projects/neural-audio-coding

TABLE IV
ABLATION ANALYSIS ON BLENDING WEIGHTS

(a) Model-I (a neural codec only)

Blending Ratio (MSE : mel)	Decoded SNR (dB)	Decoded PESQ
1 : 0	**18.12**	3.67
0 : 1	0.16	4.23
1 : 1	6.23	4.31
10 : 1	16.88	**4.37**

(b) Model-III (a neural codec with a collaboratively trained LPC)

Blending Ratio (MES : mel)	Residual SNR (dB)	Decoded SNR (dB)	Decoded PESQ
1 : 0	**9.73**	14.25	3.84
0 : 1	1.79	17.23	4.02
1 : 1	7.11	**17.82**	**4.08**
10 : 1	8.26	17.55	4.01

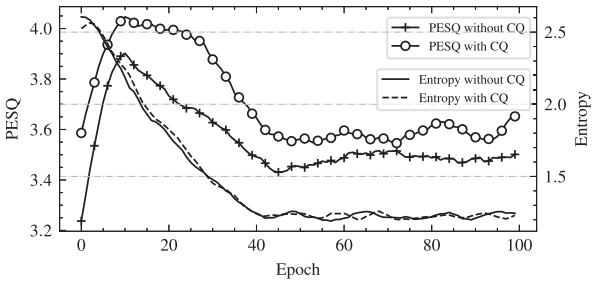


Fig. 8. The ablation analysis on CQ.

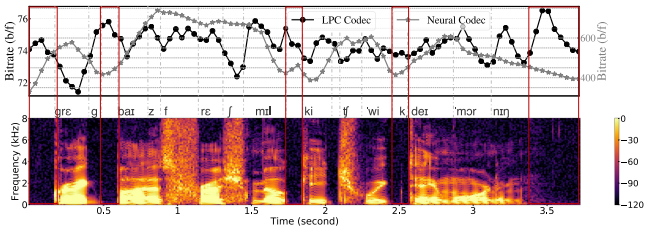


Fig. 9. The frame-wise bit allocation analysis.

TABLE V
BIT ALLOCATION AMONG CODING COMPONENTS

Bitrate Modes (kbps)	LPC Coefficients (bits / frame)	LPC Residual (bits / frame)	Total (bits / frame)
~ 11.77	58	295	353
~ 19.20	74	502	576
~ 30.72	74	486+384	944

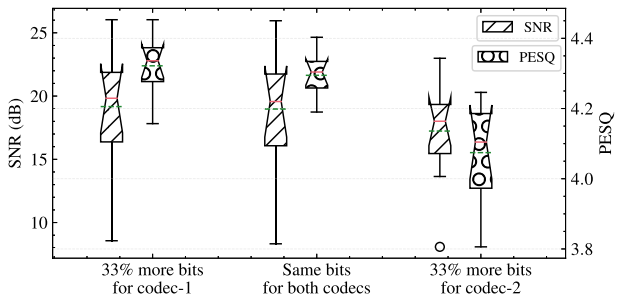


Fig. 10. Ablation analysis on bit allocation schemes between codec-1 and codec-2 in Model-IV at 32 kbps.

$\frac{512-32}{16000*1024} * 256 * 2.5 \approx 21.3$ kbps. The SNR reaches the highest when there is only the MSE term, while the PESQ score becomes the lowest. By only keeping the mel-scaled loss term, the PESQ score is decent (4.23), yet with a poor waveform reconstruction as suggested by the SNR value (0.16 dB). For Table IV (b), the target entropy for each LPC coefficient is of 5-bit or 2.6 kbps, and 3.5-bit for each LPC residual sample or 29.9 kbps. Similarly, even with the input of the neural codec being the LPC residuals, MSE alone yields the highest SNR for the reconstruction of the LPC residual, but it does not benefit the final synthesized signal even in terms of SNR. Note that we choose 128 quantization centroids for the high bitrate case, which is different from that of Model-III in Table III where 32 quantization centroids are employed. For consistency's sake, we choose the blending ratio of $10:1$, which shows reasonably well numbers in all proposed models.

2) CQ's Impact on the Speech Quality: We compare the PESQ values of the decoded signals from Model II and III. Since Model-III shares the same architecture with Model-II except for the CQ training strategy, the comparison is to verify that CQ can effectively allocate bits to the LPC and NWC modules. Fig. 8 shows that the total entropy of the two models are under the control regardless of the use of CQ mechanism. However, we can see that Model-III with CQ achieves higher PESQ during and after the control of the entropy, showcasing that the CQ approach benefits the codec's performance.

3) Bit Allocation Between the LPC and NWC Modules: Since the proposed CQ method is capable of assigning different bits to the LPC and NWC modules dynamically, i.e., in a frame-by-frame manner, we analyze its impact in more detail. In the mid-range bitrate setting, Fig. 9 shows the amount of bits assigned to both modules per frame (b/f). First of all, we observe that the dynamic bit allocation scheme indeed adjusts the LPC and NWC bitrates over time. Because of the CQ-enabled dynamic bit allocation, our system is able to compress silent frames more efficiently: by allocating just a few more bits to LPC, it saves a lot more bits from the NWC module for residual coding, as shown in the crimson-colored boxed areas in Fig. 9. However, it still requires a significant amount of bits to even represent those near-silent frames, which can be further optimized by voice activity detection. Finally, it appears that NWC is less efficient for fricatives (e.g., f and $\int$) and affricates (e.g., $t\int$). Table V shows the overall bit allocation among different modules. In the low bitrate case, it is worth noting that CQ uses 58 b/f or 1.93 kbps, differently from AMR-WB's standard, 2.4 kbps.

4) Bit Allocation Between the Two NWC Modules in Model-IV: To find the optimal bit allocation between two NWC modules, we first conduct an ablation analysis on 3 different bit allocation choices. In Fig. 10, both the SNR and PESQ scores degrade when the second NWC uses 33.3% more bits than the first one. Among these 3 choices, the highest PESQ score is obtained when the first NWC module uses 33.3% more bits. In practice, the bit allocation is automatically determined during the optimization process. In Table V, for example, the bit ratio between two NWC modules of Model-IV in the high bitrate case is about

TABLE VI
EXECUTION TIME RATIOS DURING MODEL INFERENCE (%)

Hardware	0.45M	0.35M	0.45M×2	0.35M×2
1× Tesla V100	12.49	13.38	20.69	21.12
1× Tesla K80	24.45	22.53	39.42	38.82
8× CPU cores	20.76	18.91	35.17	33.80
1× CPU core	46.88	42.44	87.38	80.21

$486 : 384 \approx 55.9 : 44.1$, in accord with the observation from the ablation analysis that the first module should use more bits.

G. Complexity and Delay

The proposed NWC model is with 0.35 million parameters, a half of which is for the decoder. Hence, in 32 kbps with two NWC modules for residual coding, the model size totals 0.7 M parameters, with the decoder size of 0.35 M. Even though our decoder is still not as compact as those in traditional codecs, it is $100\times$ smaller than a WaveNet decoder.

Aside from the model size, we investigate the codec's delay and the execution induced latency. The codec will have algorithmic delay if it relies on future samples to predict the current sample. The processing time during the encoding and decoding processes also adds up to the runtime overhead.

1) Algorithmic Delay: The delay of our system is defined by the frame size: the first sample of a frame can be processed only after the entire frame is buffered: $1024/16000 = 64$ms. Causal convolution can minimize such delay at the expense of the reduced speech quality, because it only uses past samples.

2) Analysis of the Execution Time: The execution time is another important factor to be considered for real-time communications. The bottom line is that the execution of the encoding and decoding processes is expected to be within the duration of the hop length so as not to lead to execution induced latency. For example, WaveNet codec [20] minimizes the system delay using causal convolution, but its processing time, though not reported, can be rather high as it is an autoregressive model with over 20 million parameters. Table VI lists the execution time ratio of our models. The ratio (in percentage) is defined as the execution time to encode and decode the test signals divided by the duration of those signals. Meanwhile, Kankanahalli's model requires 4.78 ms to encode and decode a hop length of 30 ms on an NVIDIA GeForce GTX 1080 Ti GPU, and 21.42 ms on an Intel Core™ i7-4970 K CPU (3.8 GHz), which amount to 15.93% and 71.40% of the execution time ratio, respectively [38]. Our small-sized models (0.45 M and 0.35 M) on both CPU (Intel Xeon Processor E5-2670 V3 2.3 GHz) and GPU run faster than Kankanahali's, while the direct comparison is not fair due to the different computing environment. The CMRL models with two NWC modules require more execution time. Note that all our models compared in this test achieved the real-time processing goal as their ratios are under 100%. The proposed NWC with 0.35 million parameters runs faster on CPUs than its predecessor [25], [26] with 0.45 million parameters, although the comparison is not consistent on GPUs, due to TensorFlow's optimization effects at runtime.

3) Implementation Notes and Limitations: While the proposed model noticeably reduces the decoding time and memory footprint compared to our previous models and the WaveNet decoder, it may not be ready to directly meet real-world requirements. Concretely, our neural codec can be implemented in a more hardware-friendly fashion, so as to allow the processor to handle multiple other tasks. One promising direction is to quantize the neural network. For example, instead of using the single-precision (32 b) floating points to represent model weights, we can represent each weight by one of 255 values (8-bit quantization), which enables much simpler arithmetic operations during inference. Furthermore, depending on the model architecture, pruning away less important weights is a sensible method to compress a network. Investigating these network compression methods for our codecs is beyond the scope of this paper, but it has been observed that quantization and pruning bring little to no degradation to neural networks for speech recognition [51]. The recently proposed PercepNet architecture also showed high-quality, real-time speech enhancement is possible using less than 5% of a CPU core for speech enhancement [52]. The current system is not causal with an algorithmic delay of 64 ms, which necessitates the employment of causal convolutions for real-time applications.

V. CONCLUDING REMARKS

Recent neural waveform codecs have outperformed the conventional codecs in terms of coding efficiency and speech quality, at the expense of model complexity. We proposed a scalable and lightweight neural acoustic processing unit for waveform coding. Our smallest model contains only 0.35 million parameters whose decoder with 0.12 million parameters is more than $160\times$ smaller than the WaveNet based codec. Having a compact design as a neural network, by incorporating a trainable LPC analyzer and residual cascading, our models reconstruct clean English speech samples with the quality on par with or superior to that from standardized codecs. Admittedly, these standardized codecs are more computationally efficient and have reasonably well performed from narrow to full band scenarios already. It is still highly desired if these standalone DSP components can be reformulated into a lightweight but end-to-end trainable format for a full neural speech processing pipeline. To that end, the proposed system serves a candidate as it operates in a frame-wise manner with the processing time less than the frame length, even with a less optimized python-based implementations. The proposed system is still computationally heavier than traditional speech codecs. Therefore, running the model in embedded systems with limited computational resources may require further model compression, such as parameter quantization and pruning. Although the neural waveform codec is generic and not contingent upon language specific priors, its generalizability to different languages and noisy and reverberant acoustic environments is not guaranteed, especially if the model sizes are relatively small.

We open-sourced the project. The source code and sound examples can be found at: https://saige.sice.indiana.edu/research-projects/neural-audio-coding.

REFERENCES

[1] J.-M. Valin and J. Skoglund, "A real-time wideband neural vocoder at 1.6 kb/s using LPCNet," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2019, pp. 3406–3410.

[2] M. R. Schroeder, "Vocoders: Analysis and synthesis of speech," *Proc. IEEE*, vol. 54, no. 5, pp. 720–734, May 1966.

[3] A. V. McCree and T. P. Barnwell, "A mixed excitation LPC vocoder model for low bit rate speech coding," *IEEE Trans. Speech Audio Process.*, vol. 3, no. 4, pp. 242–250, Jul. 1995.

[4] M. Dietz *et al.*, "Overview of the EVS codec architecture," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 5698–5702.

[5] B. Bessette *et al.*, "The adaptive multirate wideband speech codec (AMR-WB)," *IEEE Trans. Speech Audio Process.*, vol. 10, no. 8, pp. 620–636, Nov. 2002.

[6] J. Stachurski and A. McCree, "Combining parametric and waveform-matching coders for low bit-rate speech coding," in *Proc. 10th Eur. Signal Process. Conf.*, 2000, pp. 1–4.

[7] M. Schroeder and B. Atal, "Code-excited linear prediction (CELP): High-quality speech at very low bit rates," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 10, pp. 937–940, 1985.

[8] J. Stachurski and A. McCree, "A 4 kb/s hybrid MELP/CELP coder with alignment phase encoding and zero-phase equalization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2000, pp. 1379–1382.

[9] S. Bruhn *et al.*, "Standardization of the new 3GPP EVS codec," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 5703–5707.

[10] A. Rämö and H. Toukomaa, "Subjective quality evaluation of the 3GPP EVS codec," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 5157–5161.

[11] J. Skoglund and J.-M. Valin, "Improving opus low bit rate quality with neural speech synthesis," in *Proc. Annu. Conf. Int. Speech, Commun. Assoc. (Interspeech)*, 2020, pp. 2847–2851.

[12] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6306–6315.

[13] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using WaveNet autoencoders," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 12, pp. 2041–2053, Dec. 2019.

[14] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 1, pp. 7–19, Jan. 2015.

[15] Y. Luo and N. Mesgarani, "TasNet: Surpassing ideal time-frequency masking for speech separation," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 8, pp. 1256–1266, Aug. 2019.

[16] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2017, pp. 2616–2620.

[17] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process*, 2013, pp. 6645–6649.

[18] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 10, pp. 1533–1545, Oct. 2014.

[19] A. Van Den Oord *et al.*, "WaveNet: A generative model for raw audio," 2016, *arXiv:1601.06071*.

[20] W. B. Kleijn *et al.*, "WaveNet based low rate speech coding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 676–680.

[21] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2014.

[22] Y. L. C. Garbacea and A. van den Oord, "Low bit-rate speech coding with VQ-VAE and a WaveNet decoder," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2019, pp. 735–739.

[23] J.-M. Valin and J. Skoglund, "LPCNet: Improving neural speech synthesis through linear prediction," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 5891–5895.

[24] N. Kalchbrenner *et al.*, "Efficient neural audio synthesis," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2410–2419.

[25] K. Zhen, J. Sung, M. S. Lee, S. Beack, and M. Kim, "Cascaded cross-module residual learning towards lightweight end-to-end speech coding," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2019, pp. 3396–3400.

[26] K. Zhen, M. S. Lee, J. Sung, S. Beack, and M. Kim, "Efficient and scalable neural residual waveform coding with collaborative quantization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 361–365.

[27] C. Un and D. Magill, "The residual-excited linear prediction vocoder with transmission rate below 9.6 kbits/s," *IEEE Trans. Commun.*, vol. 23, no. 12, pp. 1466–1474, Dec. 1975.

[28] K. Zhen, M. S. Lee, J. Sung, S. Beack, and M. Kim, "Psychoacoustic calibration of loss functions for efficient end-to-end neural audio coding," *IEEE Signal Process. Lett.*, vol. 27, pp. 2159–2163, Nov. 2020.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[30] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 933–941.

[31] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2016.

[32] K. Tan, J. T. Chen, and D. L. Wang, "Gated residual networks with dilated convolutions for supervised speech separation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 21–25.

[33] K. Tan, J. T. Chen, and D. L. Wang, "Gated residual networks with dilated convolutions for monaural speech enhancement," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 1, pp. 189–198, Jan. 2019.

[34] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1251–1258.

[35] W. Shi *et al.*, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1874–1883.

[36] M. Garey, D. Johnson, and H. Witsenhausen, "The complexity of the generalized Lloyd-max problem (Corresp.)," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 255–256, Mar. 1982.

[37] E. Agustsson *et al.*, "Soft-to-hard vector quantization for end-to-end learning compressible representations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1141–1151.

[38] S. Kankanahalli, "End-to-end optimized speech coding with deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 2521–2525.

[39] F. Itakura, "Early developments of LPC speech coding techniques," in *Proc. Int. Conf. Spoken Lang. Process.*, 1990, pp. 1409–1410.

[40] I. R. Titze, "Principles of voice production," *Nat. Center Voice Speech*, 1994.

[41] D. O'Shaughnessy, "Linear predictive coding," *IEEE Potentials*, vol. 7, no. 1, pp. 29–32, 1988.

[42] J. Franke, "A levinson-durbin recursion for autoregressive-moving average processes," *Biometrika*, vol. 72, no. 3, pp. 573–581, 1985.

[43] ITU-T G. 722.2, "Wideband coding of speech at around 16 kbit/s using adaptive multi-rate wideband (AMR-WB)," 2003.

[44] A. Gersho and V. Cuperman, "Vector quantization: A pattern-matching technique for speech coding," *IEEE Commun. Mag.*, vol. 21, no. 9, pp. 15–21, Dec. 1983.

[45] F. Soong and B. Juang, "Line spectrum pair (LSP) and speech data compression," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 9, pp. 37–40, 1984.

[46] A. Rix, J. Beerends, M. Hollier, and A. Hekstra, "Perceptual evaluation of speech quality (PESQ)-A new method for speech quality assessment of telephone networks and codecs," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 2, pp. 749–752, 2001.

[47] ITU-R Recommendation BS 1534-1, "Method for the subjective assessment of intermediate quality levels of coding systems (MUSHRA)," 2003.

[48] J. S. Garofolo *et al.*, "TIMIT acoustic-phonetic continuous speech corpus," *Linguistic Data Consortium*, 1993.

[49] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.

[50] J.-M. Valin, G. Maxwell, T. B. Terriberry, and K. Vos, "High-quality, low-delay music coding in the Opus codec," in *Audio Eng. Soc. Convention 135*, Oct. 2013.

[51] S. Han *et al.*, "ESE: Efficient speech recognition engine with sparse LSTM on FPGA," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, New York, NY, USA: Association for Computing Machinery, 2017, pp. 75–84.

[52] J.-M. Valin, U. Isik, N. Phansalkar, R. Giri, K. Helwani, and A. Krishnaswamy, "A perceptually-motivated approach for low-complexity, real-time enhancement of fullband speech," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2020, pp. 2482–2486.

Kai Zhen (Student Member, IEEE) received the B.S. degree in software engineering from Xidian University, Xi'an, China, in 2012, the M.S. degree in computer science from Tsinghua University, Beijing, China, in 2015, and the Ph.D. degree in computer science and cognitive science from Indiana University, Bloomington, IN, USA. He is currently an Applied Scientist with Amazon. He has worked on scalable, efficient, and psychoacoustically plausible neural waveform coding for speech and audio signals. He was the recipient of the Outstanding Research Award from the IU cognitive science program for his dissertation research.

Jongmo Sung received the B.S. and M.S. degrees in electronics engineering from Pusan Nation University, Busan, South Korea, in 1995 and 1997, respectively, and the Ph.D. degree in mechatronics engineering from Chungnam National University, Daejeon, South Korea, in 2014. Since 1999, he has been a Principal Researcher with Media Coding Research Section, ETRI, Daejeon, South Korea. His research interests include a wide range of topics in speech and audio signal processing, including speech and audio compression.

Mi Suk Lee received the B.S. and M.S. degrees in electronics engineering from Hoseo University, Asan, South Korea, in 1991 and 1993, respectively, and the Ph.D. degree in electrical and electronics engineering from the Korea Advanced Institute of Science & Technology, Daejeon, South Korea, in 2001. Since Febuary 2002, she has been a Principal Researcher with ETRI, Daejeon, South Korea. Her current research interests include digital speech and audio coding, digital audio signal processing techniques for immersive broadcasting and digital twins.

Seungkwon Beack received the B.S. degree in electronic engineering from Korea Aviation University, Koyang, South Korea, in 1999, and the M.S. and Ph.D. degrees from the Department of Information and Communications Engineering, the Korea Advanced Institute of Science & Technology, Daejeon, South Korea, in 2001 and 2005, respectively. He is currently a Principal Researcher with ETRI, Daejeon, South Korea. His research interests include audio signal processing and coding.

Minje Kim (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2016. He is currently an Assistant Professor of intelligent systems engineering with Indiana University, Bloomington, IN, USA, where he is also affiliated with data science, cognitive science, and statistics. He is also an Amazon Visiting Academic. Before that from 2006 to 2011, he was a Researcher with ETRI, Daejeon, South Korea. He is a Member of the IEEE AASP TC. He was the recipient of the NSF CAREER Award (2021), IEEE SPS Best Paper Award (2020), Google and Starkey's grants for outstanding student papers at ICASSP 2013 and 2014, respectively.