NEURAL FEATURE PREDICTOR AND DISCRIMINATIVE RESIDUAL CODING FOR LOW-BITRATE SPEECH CODING

Haici Yang[1], *Wootaek Lim*[2], *Minje Kim*[1]

[1]Indiana University, Luddy School of Informatics, Computing, and Engineering, Bloomington, IN, USA
[2] Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea

ABSTRACT

Low and ultra-low-bitrate neural speech codecs achieved unprecedented coding gain by generating speech signals from compact features. This paper introduces additional coding efficiency in speech coding by reducing the temporal redundancy existing in the frame-level feature sequence via a feature predictor. This predictor produces low-entropy residual representations, and we discriminatively code them based on their contribution to the signal reconstruction. Combining feature prediction and discriminative coding optimizes bitrate efficiency by assigning more bits to hard-to-predict events. We demonstrate the advantage of the proposed methods using the LPCNet as a neural vocoder, resulting in a scalable, lightweight, low-latency, and low-bitrate neural speech coding system. While our approach guarantees strict causality in the frame-level prediction, the subjective tests and feature space analysis show that our model achieves superior coding efficiency compared to the loosely-causal LPCNet and Lyra V2 in the very low bitrates.

Index Terms— Low-bitrate Speech Codec, Predictive Coding, Generative Model, LPCNet

1. INTRODUCTION

A speech codec, in general, comprises modules for speech compression, quantization, and reconstruction. It has been used in various communication and entertainment applications after standardization [1, 2] and open-sourcing [3]. The common goal in speech coding is to achieve the maximum coding gain, i.e., maintaining the perceptual quality and intelligibility of the reconstructed speech signals with a minimum bitrate.

The involvement of neural networks has greatly benefited the coding trade-off, effectively eliminating the codes' redundancy while improving the reconstruction quality. More recently, the advances of generative models and their applications in speech coding led to a trend in very low-bitrate speech codecs. The first WaveNet-based speech codec [4] demonstrates the usage of neural synthesis in both waveform and parametric coding. The latter is more favored in subsequent studies because of its inherent advantages in dealing with very condensed speech features. These neural vocoders work on the decoder side, leveraging the powerful neural synthesis architecture. Their encoding parts are relatively simplified, relying on existing Codec 2 codes [5] as in the original WaveNet-based speech codec [4] or the dimension-reduced frequency-domain speech representations, e.g., cepstrum features [6, 7], and LPC analysis [8].

In this line of work, the performance bottleneck comes from the very compact codes. Some efforts apply more complex models in the encoder to improve the quality of the features [9, 10] or put generative models [11, 12, 13] at the end of the existing codec to facilitate signal restoration by post-processing. However, the output performance is still bounded by the quality of the coding features.

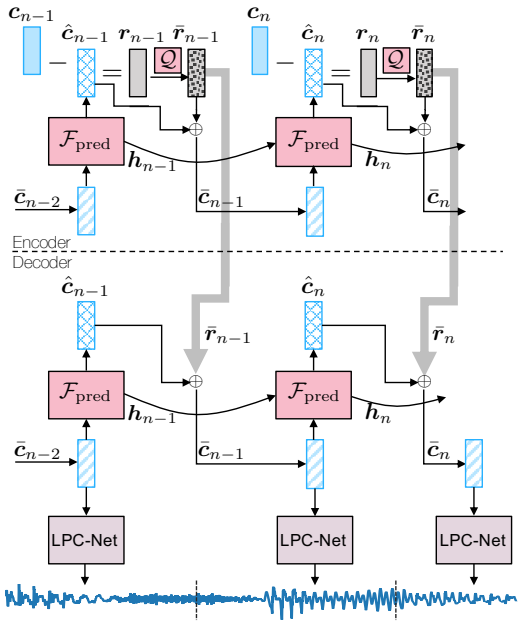

Fig. 1: The overview of the proposed neural speech coding system with feature predictors and LPCNet-based vocoder.

End-to-end neural codecs that train the encoder, quantizer, and decoder jointly work as an alternative to the low-bitrate generative speech vocoders [14, 15, 16, 17]. In this way, the neural encoder participates in removing the redundancy in the source signal and produces features that are more associated with the decoder, in contrast to the traditional speech features. Various other methods have been developed to improve the quality of the features, regarding robustness [18, 19], scalability [20] and the variability issues [21].

However, end-to-end codecs tend to suffer in very low-bitrates cases (<2 kbps) because that requires the coding features to be extremely small and expressive simultaneously. To deal with that, an ultra-low bitrate codec [22] borrows the embeddings from a self-supervised training task to increase the expressiveness of the state-of-the-art codec SoundStream's features [17], and can obtain a decent speech quality with a very low bitrate, 0.6 kbps. TF-Codec [23] addresses the problem by reducing the temporal redundancy in the latent features with a predictive model and reports decent reconstruction quality at 1 kbps. However, both models entail high complexity. Besides, because TF-Codec's prediction model runs on a latent space

This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government (23ZH1200; "The research of the basic media contents technologies").

that requires a specific pair of encoder and decoder, it brings an extra cost for other existing codecs to mount its predictive module directly.

In this paper, we aim at a low-bitrate, low-delay, and low-complexity neural speech codec that utilizes neural feature prediction to reduce the temporal redundancy from the sequence of feature frames. We introduce a gated recurrent unit (GRU)-based [24] frame-level feature predictor that can forecast the feature vector at time frame t using its preceding frames. Since the decoder also employs the exact feature predictor, it can "generate" most of the feature vector at no cost of bitrate, while the imperfectly generated feature vectors are compensated by the coded residual sent from the encoder side. Additionally, we employ *discriminative coding* in the residual space. This idea is demonstrated in source-aware neural audio coding by distinguishing speech and noise sources in the latent feature space [25]. In this paper, we use different entropy coding strategies at each frame depending on the amount of information they carry. Compared to the TF-Codec, our model explicitly codes only the prediction residuals, and the proposed predictive modules are designed to work in combination with existing low-complexity neural codecs. In particular, we base the predictive module on the efficient LPCNet-based speech coding framework [7], and predicts mainly cepstral coefficients.

2. THE PROPOSED PREDICTIVE CODING

2.1. Overview

In conjunction with the LPCNet's sample-level vocoding, the proposed feature prediction model performs hierarchical prediction: first in the feature space and then in the sample level. As shown in Fig 1, the frame-level feature predictor $\mathcal{F}_{pred}$ works autoregressively on both the encoder and decoder sides. The encoder computes and quantizes the frame-level prediction residuals $\bar{\boldsymbol{r}}$ and passes them to the decoder. Then, the decoder adds the received residuals to its own feature predictions $\hat{\boldsymbol{c}}$ to obtain the recovered frame-level features. The sample-level predictive coding (i.e., the LPCNet vocoder)synthesizes waveform samples $\hat{\boldsymbol{s}}$ from the recovered features as the codec's output.

2.2. The frame-level feature predicion

2.2.1. Feature predictor

We apply a WaveRNN-based model [26] to make frame-level predictions on the 18-dimensional continuous cepstral coefficients. WaveRNN explicitly considers the output at time $n-1$ as the estimation of the n-th's sample. In our frame-by-frame feature prediction scenario, the recurrent neural network $\mathcal{H}(\cdot)$ takes in previous hidden state $\boldsymbol{h}_{n-1}$ and the previous feature vector $\boldsymbol{c}_{n-1}$, to predict the next frame $\hat{\boldsymbol{c}}_n$. Additionally, we condition the frame-level prediction with pitch parameters $\boldsymbol{m}_n$ (period and correlation) used in LPCNet. Our model consists of two gated recurrent unit (GRU) layers [24], with 384 and 128 hidden units, respectively, followed by a fully connected layer. The feature predictor $\mathcal{F}_{\text{pred}} : \boldsymbol{c}_{n-1} \mapsto \hat{\boldsymbol{c}}_n$ can therefore be recursively defined as,

$$\boldsymbol{h}_n = \mathcal{H}(\boldsymbol{c}_{n-1}, \boldsymbol{h}_{n-1}, \boldsymbol{m}_n), \quad \hat{\boldsymbol{c}}_n = \tanh(\boldsymbol{W}\boldsymbol{h}_n), \tag{1}$$

where n represents the time-domain frame index. We found the results are more stable by scaling input and output features to the range of $[-1, 1]$. To this end, the output linear layer employs a tanh activation after a linear combination with parameter $\boldsymbol{W}$. Biases are omitted for brevity.

We optimize the model by minimizing the mean squared error (MSE) between the prediction and target $\mathcal{L} = MSE(\boldsymbol{c}_n, \hat{\boldsymbol{c}}_n)$. We chose it over the maximum log-likelihood approach and explicit Gaussian modeling because modeling the cepstrum coefficients with Gaussian distributions was unreliable and we found that the MSE loss leads to more stable prediction.

2.2.2. Feature residual coding

The predictor operates in both the encoder and decoder to cover the information that are inferable from the temporal dependency. Thus, for the decoder to recover the features, it is only necessary to provide the decoder with the residuals between the prediction and ground-truth features. This explicit residual coding can lead to a more efficient coding scheme by reducing the code's entropy, given that our predictor model makes reliable predictions, especially in the areas of smooth signals.

The primary pipeline for residual coding is then summarized recursively as follows:

$$\text{Encoder:} \quad \hat{\boldsymbol{c}}_n = \mathcal{F}_{\text{pred}}(\bar{\boldsymbol{c}}_{n-1}) \tag{2}$$

$$\boldsymbol{r}_n = \boldsymbol{c}_n - \hat{\boldsymbol{c}}_n \tag{3}$$

$$\bar{\boldsymbol{r}}_n = \mathcal{Q}(\boldsymbol{r}_n) \quad \text{(send it to the decoder)} \tag{4}$$

$$\bar{\boldsymbol{c}}_n = \hat{\boldsymbol{c}}_n + \bar{\boldsymbol{r}}_n \quad \text{(input for the next round } n+1) \tag{5}$$

The encoder explicitly computes the residual $\boldsymbol{r}_n$, and then the quantizer $\mathcal{Q}(\cdot)$ converts it into a bitstring $\bar{\boldsymbol{r}}_n$ as the final code. Note that we opt to input the *noisy feature* $\bar{\boldsymbol{c}}$ instead of the original feature $\boldsymbol{c}$ into the encoder's feature predictor (eq. (2)) in order to match the encoder's output to the decoder's circumstance. Since the decoder does not have access to the original features, it has no choice but to use the noisy ones $\bar{\boldsymbol{c}}$ as the predictor's input. Therefore, by repeating the decoder's behavior in the encoder, we aim to guarantee that the residuals provided by the encoder are the accurate compensation for the decoder's feature prediction.

The decoder first pre-computes the prediction $\hat{\boldsymbol{c}}_n$, and then supplements it with the quantized residual $\bar{\boldsymbol{r}}_n$ received from the encoder to finalize the feature reconstruction $\bar{\boldsymbol{c}}_n$.

$$\text{Decoder:} \quad \hat{\boldsymbol{c}}_n = \mathcal{F}_{\text{pred}}(\bar{\boldsymbol{c}}_{n-1}) \tag{6}$$

$$\bar{\boldsymbol{c}}_n = \hat{\boldsymbol{c}}_n + \bar{\boldsymbol{r}}_n. \tag{7}$$

When running $\mathcal{F}_{\text{pred}}$ in either the encoder or decoder, we starts with zero-initialized input $\bar{\boldsymbol{c}}_0 = \boldsymbol{0}$, and iteratively update the input tensor with the model predictions .

2.3. The sample-level vocoder: LPCNet

We borrow LPCNet to complete time-domain synthesis on the decoder side. The inputs to the LPCNet are the quantized pitch parameters $\boldsymbol{m}_n$ and the 18-band Bark-frequency cepstrum features $\boldsymbol{c}_n$. LPCNet estimates the prediction coefficients a_τ from the input cepstrum and integrates LPC analysis, i.e., $p_t = \sum_{\tau=1}^{T} a_\tau \hat{s}_{t-\tau}$, into the neural generative model to take away the burden of envelop modeling. In addition to the above DSP-based linear prediction, LPCNet employs a WaveRNN network $\mathcal{G}$ to focus on estimating the prediction residuals e_t in a causal manner:

$$\hat{e}_t = \mathcal{G}(p_t, \hat{s}_{<t}, \hat{e}_{<t}), \quad \hat{s}_t = p_t + \hat{e}_t, \tag{8}$$

where the quality of the estimated excitation signal $\hat{e}_t$ significantly matters for better speech quality. The network $\mathcal{G}$ mainly consists of two GRU layers, followed by two fully connected layers.

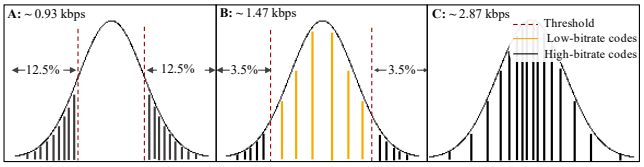

Fig. 2: The proposed thresholding mechanism uses different quantization schemes depending on the target bitrate.

We employ the same LPCNet vocoder for our coding system, although replacing its input feature $\boldsymbol{c}_n$ with our proposed feature reconstruction $\bar{\boldsymbol{c}}_n$ necessitates re-training the vocoder. Note that the speech coding-oriented [7] LPCNet also take its own compressed feature representation as input, whose compression ratio is something our method competes against.

3. THE PROPOSED DISCRIMINATIVE RESIDUAL CODING

While the predictive module saves bitrates by sending residual signals $\bar{\boldsymbol{r}}_n$ to the receiver in place of the full cepstrum $\boldsymbol{c}_n$, applying *discriminative* coding to the residuals $\bar{\boldsymbol{r}}_n$ improves the coding gain further. Due to the overall smoothness of speech, the prediction in the cepstrum domain results in the residuals that follow a Gaussian distribution with zero mean and small variance. The larger residual values mainly occur in transient events, such as plosives. To fully make use of the residual signal's statistical advantage, we apply *discriminative* coding that distinguishes the more "code-worthy" frames from the rest by thresholding the L_1 norm of the residuals. This way, frames with significant residual energy are assigned more bits, and bits assigned to the less significant frames are minimized.

Scanning the entire training set, we define a threshold value θ depending on the target bitrate. The quantization process eq.(4) is therefore expanded to:

$$\bar{\boldsymbol{r}} = \mathcal{Q}(\boldsymbol{r}) = \begin{cases} \mathcal{Q}_{\mathrm{L}}(\boldsymbol{r}) & \text{if } ||\boldsymbol{r}||_1 \geq \theta \\ \mathcal{Q}_{\mathrm{S}}(\boldsymbol{r}) & \text{otherwise,} \end{cases} \tag{9}$$

with $\mathcal{Q}_{\mathrm{L}}$ and $\mathcal{Q}_{\mathrm{S}}$ representing quantization schemes with large and small L_1 norms that use large and small codebooks, respectively. Particularly, when the target bitrate is extremely low, we *discard* low L_1 norm frames entirely, i.e., $\mathcal{Q}_{\mathrm{S}}(\boldsymbol{r}) = \boldsymbol{0}$.

The thresholding mechanism is illustrated in Fig.2. The low-bitrate scheme ($\sim$ 0.93 kbps) uses $\mathcal{Q}_{\mathrm{L}}$ for the top 25 % residual frames while discarding the rest without any coding. The intermediate bitrate ($\sim$ 1.47 kbps) case keeps the top 7% for the $\mathcal{Q}_{\mathrm{L}}$ quantization and the rest 90% for $\mathcal{Q}_{\mathrm{S}}$. The $\sim$ 2.87 kbps case uses $\mathcal{Q}_{\mathrm{L}}$ quantization for all residual frames with no thresholding.

Similar to LPCNet's coding scheme, we separately code the first component $\boldsymbol{r}_0$ of the residuals vector and the rest of dimensions $\boldsymbol{r}_{1-17}$; Note here that we dropped the frame index n and used subscript to indicate one of the 18 cepstrum coefficients within the frame. Also, since we noticed that $\boldsymbol{r}_0$ and $\boldsymbol{r}_{1-17}$ have different L_1 norm distributions, we define thresholds and apply discriminative coding to the scalar and vector components independently.

Table 1 summarizes our recommended strategies of conducting discriminative and multi-stage quantization on the different target bitrate. For scalar quantization, we use the same codebook size of $2^8 = 512$ in all $\mathcal{Q}_{\mathrm{L}}$ cases, while only 16 codewords for $\mathcal{Q}_{\mathrm{S}}$ in the mid-bitrate case or skips coding in the low-bitrate case. All scalar quantizers use a single-stage quantization scheme. As for the VQ for $\boldsymbol{c}_{1:17}$, we employ either one or two-stage quantization for $\mathcal{Q}_{\mathrm{L}}$ with a codebook of size 1024 in each stage; $\mathcal{Q}_{\mathrm{S}}$ cases use a single 512-size codebook or skip coding in the low-bitrate case ($\sim$0.93). We also estimate the bitrate considering Huffman coding by computing the frequencies $\boldsymbol{p}$ of all codewords from coding randomly-selected 2-second segments per training samples and derive the average bit-per-frame by $-\sum_i^N \boldsymbol{p}_i log_2 \boldsymbol{p}_i$. Apart from the bits we have stated in the Table 1, we also need to count in the bits for coding pitch parameters in LPCNet's original way, which takes up 0.275 kbps. We use the bitrates of Huffman coding in the rest of the paper, although it is close to the bitrates based on the codebook. In the $\sim$0.93kbps case, for example, the target bitrate in our table is calculated by $0.25\times(7\times 100+(9.9+9.8)\times 100)+275 = 932$bps, given that each frame is for 10 ms (meaning 100 frames per second), and only 25% of the frames are coded in this example. We consider the thresholds as parameters that are pre-defined in the model building stage for different bitrate ranges, thus need not to be included in bitrate calculation.

Target bitrate (kbps)	$\sim$ 0.93		$\sim$ 1.47		$\sim$ 2.87	
$\mathcal{Q}_{\mathrm{L}}$ percentage	25 %		7 %		100 %	
Codebook Size (bits) : Bits-per-frame according to Huffman coding						
Stages	1st	2nd	1st	2nd	1st	2nd
$\mathcal{Q}_{\mathrm{L}}(\boldsymbol{r}_0)$	8 : 7.0	-	8 : 7.4	-	8 : 7.2	-
$\mathcal{Q}_{\mathrm{S}}(\boldsymbol{r}_0)$	-	-	4 : 2.9	-	-	-
$\mathcal{Q}_{\mathrm{L}}(\boldsymbol{r}_{1:17})$	10 : 9.8	10 : 9.9	10 : 9.2	10 : 9.4	10 : 9.2	10 : 9.6
$\mathcal{Q}_{\mathrm{S}}(\boldsymbol{r}_{1:17})$	-	-	9 : 8.0	-	-	-

Table 1: Codebook sizes and bitrate assignments.

4. EXPERIMENTS

4.1. Data

We use the Librispeech [27] corpus's `train-clean-100` fold for training, and `dev-clean` for validation, at 16kHz sampling rate. 18 Bark-scale cepstral coefficients are produced for each 20 ms frame with an overlap of 10ms. In addition, we extract and quantize the 2-dimensional pitch parameters using LPCNet's open-sourced framework.

4.2. Training

The training process consists of three steps and is conducted sequentially: prediction model training, codebook learning, and vocoder training. Hence, the results from the preceding steps will be used in the following training. Compared to a potential end-to-end learning approach, our modularized learning can circumvent the issue of dealing with non-differentiable quantization.

Both the feature predictor and the vocoder will eventually operate in a synthesis mode, where the inputs to the model are the synthesized results from the previous step. Therefore, we add noise to the input during training for a more robust development, as suggested in [7, 28]. Finally, the vocoder is finetuned with the quantized input features.

Codebook training is based on the residuals $\boldsymbol{r}$ produced from the encoder of the feature predictor $\mathcal{F}_{pred}$. For both vector and scalar codebooks, we run k-means clustering and pick the learned centroids as the codewords. When generating residuals for codebook training, the encoder skips the quantization step (eq. (4)) but will consider the residual thresholding. That is, the residual $\boldsymbol{r}_n$ will be added back to the prediction result $\hat{\boldsymbol{c}}_n$ (as in eq. 5) only if $||\boldsymbol{r}||_1 \geq \theta$. We randomly pick 2-second segments from each utterance in training set to generate the residual vector for codebook training. Codebooks are trained exclusively for each bitrate.

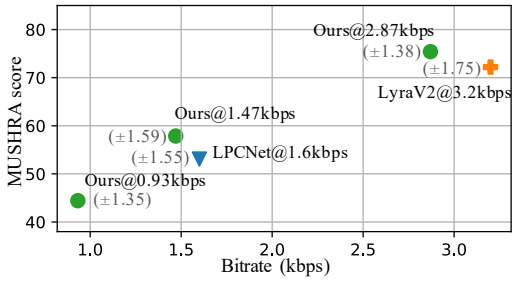

Fig. 3: Mean and 95% confidence interval of MUSHRA scores. The reference at MUSHRA score 100 is not shown in the graph.

We preserve the LPCNet's strength on low-complexity in terms of floating-point operations per second (GFLOPS) on GPUs. The total complexity of our proposed model is around 3.7 GFLOPS with increased complexity of about 0.9 GFLOPS compared to the LPCNet's original 2.8 GFLOPS. Our codec is suitable for the real-time coding task because of the causality preserved in the frame-level prediction. The algorithmic delay of our codec is 75 ms, to which the LPCNet vocoder contributes 60ms-latency from its convolution operation. Another 15 ms-delay comes from our feature predictor, which occurs while waiting for the ground-truth cepstral-frame of 10ms with an extra 5ms look-ahead to compute a cepstrum.

4.3. Evaluation and baseline

We employ two state-of-the-art low-bitrate codecs as baselines, LPCNet at 1.6kbps and Lyra V2 [1] at 3.2kbps. Lyra V2 is an improved version of Lyra[2] [21], integratin SoundStream [17] in its original architectures for a better coding gain.

In line with previous works [4], we realize that current objective metrics cannot adequately reflect the real perceptual quality of these models, so we perform a MUSHRA-style test [29] as an alternate. We use ten randomly selected gender-balanced clean utterances from the LibriSpeech `test-clean` set. Aside from the tested codecs, each trial also contains a hidden reference and a 3.5kHz low-pass-filtered anchor. Ten speech experts participated in the test, and no one was excluded per the listener exclusion policy.

5. RESULTS

5.1. Subjective test results

Fig. 3 shows the mean scores and their 95% confidence intervals from the MUSHRA-style test. Our codec outperforms LPCNet, despite using a lower bitrate ($\sim$ 1.47 vs. 1.6 kbps). At $\sim$ 0.93kbps, it is slightly worse than LPCNet at 1.6 kbps, but is acceptable given the 40% of bitrate reduction. We also achieve better perceptual quality at 2.87 kbps in comparison with Lyra V2 at 3.2kbps. This results demonstrate the proposed model's scalability and effectiveness across different bitrate ranges.

5.2. Analysis on the discriminative residual coding

In Fig. 4, we picked a random utterance sample and aligned its cepstra with the coded version of the cepstra. In between, it lays the L_1 norm curves of the scalar and vector residuals in blue and red. At

[1]https://opensource.googleblog.com/lyra-v2-a-better-faster-and-more-versatile-speech-codec.html

[2]https://ai.googleblog.com/lyra-new-very-low-bitrate-codec-for.html

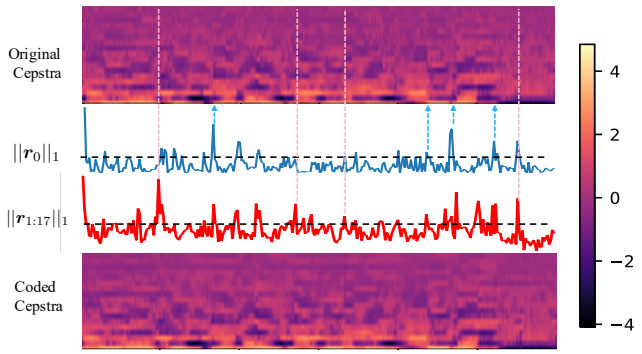

Fig. 4: The original cepstra, coded cepstra, the $L1$ norm curve for the vector residuals (red), and the L_1 norm curve for the scalar residuals (blue) of a 3s sample. The black dash lines mark the thresholds for the 1.48 kbps case.

the points where any curve is over the black dash threshold, the predictor failed to make a good prediction, thus requiring more bits to represent these residuals using a $\mathcal{Q}_{\text{L}}$ scheme. Those properly coded frames take up only 7% of the total frames in this $\sim$ 1.48 kbps case. As for the below-threshold area, conversely, the quantization falls back to the $\mathcal{Q}_{\text{S}}$ mode while being frugal in assigning bits to these well-predictable frames.

We made three main observations from the graph. Firstly, our codec does a good job estimating and coding the original cepstra (by comparing the top and bottom cepstra), especially at the lower dimensions, although with some energy loss at the higher dimension. Even at the low-bitrate coding area, the cepstral patterns are still well captured. Secondly, we observe that the curves' peaks align well with the original cepstrum's transient events. To address the alignment, we marked some arrows and dash lines that connect the residual L_1 norm curves to the cepstra as examples. The alignment of cepstral changes and the peaks ensures that, although the performance of the feature predictor degrades at the transient events, the discriminative coding can make accurate compensation at a minimal cost. Finally, it is also noteworthy that despite embracing similarity, the L_1 norm of the vector and scalar can have peaks and valleys at different places. Hence, they could compensate for the predictor model's different behavior at individual subbands, which also justifies the advantage of band-specific discriminative coding for the SQ and VQ parts.

6. CONCLUSION

In this work, we proposed a lightweight, low-latency, low-bitrate speech coding framework. In line with the parametric coding paradigm, we designed a feature predictor to capture the temporal redundancy and reduce the burden of coding raw feature frames. Moreover, we applied the discriminative coding scheme to the residual signal to further improve coding gain. We showed that the proposed combination of predictive coding and discriminative residual coding can be harmonized well with the original LPCNet-based codec by providing a more effective quantization scheme than the original multi-stage VQ. We open-source our codes at `https://saige.sice.indiana.edu/research-projects/predictive-LPCNet`.

7. REFERENCES

[1] B. Bessette *et al.*, "The adaptive multirate wideband speech codec (AMR-WB)," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 8, pp. 620–636, 2002.

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

[3] J. M. Valin, K. Vos, and T. Terriberry, "Definition of the Opus audio codec," *IETF, September*, 2012.

[4] W. B. Kleijn *et al.*, "WaveNet based low rate speech coding," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018, pp. 676–680.

[5] D. Rowe, "Codec 2 - open source speech coding at 2400 bits/s and below." 2011. [Online]. Available: http://www.tapr.org/pdf/DCC2011-Codec2-VK5DGR.pdf

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

[7] ——, "A real-time wideband neural vocoder at 1.6 kb/s using LPCNet," in *Proc. Interspeech*, 2019.

[8] J. Klejsa, P. Hedelin, C. Zhou, R. Fejgin, and L. Villemoes, "High-quality speech coding with SampleRNN," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2019.

[9] H. Kim, J. Yoon, W. Cho, and N. Kim, "Neurally optimized decoder for low bitrate speech codec," *IEEE Signal Processing Letters*, vol. 29, pp. 244–248, 2021.

[10] T. Yoshimura, K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, "WaveNet-based zero-delay lossless speech coding," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 153–158.

[11] 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, 2018.

[12] J. Skoglund and J.-M. Valin, "Improving Opus low bit rate quality with neural speech synthesis," *arXiv preprint arXiv:1905.04628*, 2019.

[13] A. Biswas and D. Jia, "Audio codec enhancement with generative adversarial networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 356–360.

[14] S. Kankanahalli, "End-to-end optimized speech coding with deep neural networks," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018.

[15] 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. Interspeech*, 2019.

[16] ——, "Scalable and efficient neural speech coding: A hybrid design," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 12–25, 2022.

[17] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "Soundstream: An end-to-end neural audio codec," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 30, p. 495–507, jan 2022.

[18] J. Casebeer *et al.*, "Enhancing into the codec: Noise robust speech coding with vector-quantized autoencoders," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 711–715.

[19] F. Lim, W. Kleijn, M. Chinen, and J. Skoglund, "Robust low rate speech coding based on cloned networks and WaveNet," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020, pp. 6769–6773.

[20] X. Jiang, X. Peng, H. Xue, Y. Zhang, and Y. Lu, "Cross-scale vector quantization for scalable neural speech coding," *arXiv preprint arXiv:2207.03067*, 2022.

[21] W. Kleijn *et al.*, "Generative speech coding with predictive variance regularization," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6478–6482.

[22] A. Siahkoohi, M. Chinen, T. Denton, W. Kleijn, and J. Skoglund, "Ultra-low-bitrate speech coding with pretrained transformers," in *Proc. Interspeech*, 2022.

[23] X. Jiang, X. Peng, H. Xue, Y. Zhang, and Y. Lu, "Predictive neural speech coding," *arXiv preprint arXiv:2207.08363*, 2022.

[24] K. Cho *et al.*, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[25] H. Yang, K. Zhen, S. Beack, and M. Kim, "Source-aware neural speech coding for noisy speech compression," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2021.

[26] N. Kalchbrenner *et al.*, "Efficient neural audio synthesis," in *Proc. of the International Conference on Machine Learning (ICML)*, vol. 80, 2018, pp. 2410–2419.

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

[28] Z. Jin, A. Finkelstein, G. J. Mysore, and J. Lu, "FFTNet: A Real- Time Speaker-Dependent Neural Vocoder," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018, pp. 2251–2255.

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