Post-Training Latent Dimension Reduction in Neural Audio Coding

Thomas Muller, Stéphane Ragot, Pierrick Philippe, Pascal Scalart

Post-Training Latent Dimension Reduction in Neural Audio Coding

Thomas Muller
Orange Innovation
IRISA, University of Rennes
Lannion, France
thomas.muller@orange.com

Stéphane Ragot
Orange Innovation
Lannion, France
stephane.ragot@orange.com

Pierrick Philippe
Orange Innovation
Rennes, France
pierrick.philippe@orange.com

Pascal Scalart
IRISA, University of Rennes
Lannion, France
pascal.scalart@irisa.fr

***Abstract*—This work addresses the problem of latent space quantization in neural audio coding. A covariance analysis of latent space is performed on several pre-trained audio coding models (Lyra V2, EnCodec, AudioDec). It is proposed to truncate latent space dimension using a fixed linear transform. The Karhunen-Loève transform (KLT) is applied on learned residual vector quantization (RVQ) codebooks. The proposed method is applied in a backward-compatible way to EnCodec, and we show that quantization complexity and codebook storage are reduced (by 43.4%), with no noticeable difference in subjective AB tests.**

***Index Terms*—Neural audio coding, vector quantization, latent space, Karhunen-Loève transform**

I. Introduction

Over the past few years, a new generation of audio codecs has emerged using neural networks. These methods target various applications, such as voice/video calling [1], [2], text-to-speech synthesis [3], or music generation from text prompts [4], [5]. Neural audio coding has made significant progress, in particular by leveraging development of generative audio models [6]–[10], autoencoders with discretized latent space [11], and advanced architectures based on generative adversarial networks (GAN) [12], [13].

While some early codecs such as LPCNet [10] and Lyra [1] demonstrated promising performance with real-time operation, we focus in this work on more recent codecs such as SoundStream [14], Lyra V2 [2], EnCodec [15] and AudioDec [16] which share similar principles with an autoencoder or vocoder trained in a GAN framework. These codecs map the input signal into latent space which is quantized by residual vector quantization (RVQ). RVQ was introduced in [14] and reused in [15], [16]. RVQ is a kind of multistage VQ [17] with several stages of unstructured codebooks.

Computational complexity depends on codec operation points, implementation platforms and optimizations, however it was observed that recent neural audio codecs often require more computation power than traditional codecs such as EVS or Opus [14]. Regarding memory requirements, EVS uses about 150 kWords of RAM, 150 kWords of (table) ROM, and more than 100k program instructions [18]; in neural models considered here, storage is significantly higher, with typically from 10 to 50M model parameters with around 4M parameters for RVQ codebooks. It is therefore of interest to consider reducing complexity and memory requirements in neural audio coding; one possible, hybrid, approach is to include classical signal processing methods in neural models.

In this paper we investigate how to optimize RVQ for latent space quantization; the proposed optimization is performed in a post-training approach, i.e., based on learned codebooks from pre-trained models. This approach avoids issues of dataset dependence and allows optimizing existing models in a way consistent with the underlying training, even with no access to the actual training dataset. We observe that there is a significant amount of correlation in latent space, using available pre-trained codec implementations (Lyra V2, EnCodec, AudioDec). To decorrelate latent vectors and reduce the effective quantization dimension we use the Karhunen-Loève transform (KLT) based on learned codebooks. KLT is a well-known method used for decorrelation, the transform is signal adaptive and typically based on a covariance analysis [19].

Different studies have already considered the issue of latent space dimensionality in neural audio models. There are lots of techniques (e.g., PCA [20], t-SNE [21]) to visualize latent space by dimensionality reduction into 2 or 3 dimensions; however, this is usually meant for visualizing and interpreting embeddings. The dimensionality of the latent bottleneck in an autoencoder is a hyperparameter, which may be tuned empirically by grid search [22]. A singular value decomposition (SVD) on the latent space vectors of a variational autoencoder (VAE) has been used in [23]. In the present work, we also conduct a post-training analysis of the latent space, however the purpose is not to control reconstruction fidelity vs. compactness in music synthesis, but to reduce the complexity and the storage of existing, pre-trained, neural audio codecs. Alternatively, one may modify the neural model; for instance, PCA autoencoder and information-ordered bottlenecks were proposed in [24], [25] to organize latent space (in image processing). An extra layer may be added to an autoencoder [26], directly integrating the idea of reducing latent space dimension by a "projection" stage. However, such alternatives go beyond the post-training optimization considered here.

The main contributions of this paper are listed below:

- We use learned RVQ codebooks to estimate the covariance matrix of the discretized latent representation

learned during training; we analyze the latent space learned by the encoder and observe that there is a significant amount of correlation;
- We propose a method to optimize RVQ by truncating quantization dimension, and we evaluate in details the proposed method for EnCodec.

This article is organized as follows. Section II reviews the basic architecture of a neural audio codec. Section III considers the analysis of the latent space. Section IV presents a method to optimize RVQ. Finally, Section V evaluates the proposed method before concluding in Section VI.

II. Review of neural audio coding and Latent Space Quantization

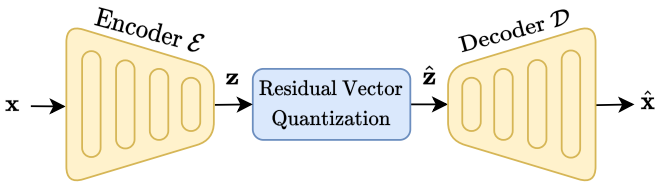

Fig. 1. Neural audio codec with RVQ (inference only).

In this work we focus on codecs using an autoencoder architecture with GAN-based training. The training phase is not covered here, more details can be found in [14]–[16].

In the inference phase, the codec comprises an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$ which are typically symmetrical, and mainly composed of 1D causal convolution layers with residual blocks to help the gradient flow through the network. Each audio frame of L samples is fed to the encoder which produces a d-dimensional latent vector $\mathbf{z}$ using strided convolutions and a last convolution layer with d output channels. After quantization, the decoder reconstructs a waveform from the quantized version of the latent vector $\hat{\mathbf{z}}$. Note that, in EnCodec, recursive layers are also included to handle long-term dependencies. As an example, EnCodec operates on mono audio sampled at $f_s = 24$ kHz with a frame length of $L = 320$ samples and a latent space of dimension $d = 128$.

Neural audio codecs studied in this work are all based on the same quantization method called Residual Vector Quantization (RVQ) in [14], which can be seen as a form of multistage (or cascaded) VQ [17]. As shown in Fig. 2, Q_1 quantizes the input latent vector $\mathbf{e}_0 = \mathbf{z}$ into $\mathbf{q}_1$ using codebook $\mathcal{C}_1$; the following stages (Q_2 to Q_N) quantize the residual $\mathbf{e}_i = \mathbf{e}_{i-1} - \mathbf{q}_i$ using codebooks $\mathcal{C}_i$. RVQ codebooks $\mathcal{C}_i$ are learned during training following the procedure of [14]. This quantization method allows for a variable bitrate by adapting the number N of stages. For instance, in EnCodec, each VQ has a codebook of 1024 codewords represented with $B = 10$ bits, and the maximum number of stages is $N_{max} = 32$, corresponding to bitrates up to 24 kbps with 0.75 kbps steps.

Table I summarizes parameters for codecs considered in this work. There is no available implementation for SoundStream [14], this codec is listed for information, given that Lyra V2 is an optimized version of SoundStream.

TABLE I
Parameters of several neural audio codecs.

Codec	f_s (kHz)	L	d	B	bitrate (kbps)	N_{max}
SoundStream	24	320	256	10	3 – 18	24
Lyra V2	16	320	64	4	3.2, 6, 9.2	46
EnCodec	24	320	128	10	1.5, 3, 6, 12, 24	32
AudioDec	24 / 48	300*	64	10	6.4 / 12.8	8

$*$ A frame length L of 320 is also available (not considered here).

III. Latent space analysis

A preliminary step in this work is to analyze the latent space learned by the encoder and quantizer. The typical approach to analyze the distribution of latent vectors generated by the encoder would be to define an audio dataset (speech, music or a mix of different audio contents) and collect vectors after encoding. This implies that results could be dataset dependent, with a potential mismatch compared to the training dataset used for the pretrained model we want to optimize. We propose in this work to use the learned codebooks which represent the latent space distribution learned during the end-to-end training. The benefit of this approach is that the analysis can be applied in a post-training phase by relying only on pre-trained codebooks.

To estimate the covariance matrix $\mathbf{R} \in \mathbb{R}^{d\times d}$ of the discretized latent space, we form all combinations $\mathbf{c}_1^{(i_1)} + \mathbf{c}_2^{(i_2)} + \cdots + \mathbf{c}_{N_{cov}}^{(i_{N_{cov}})}$ where $\mathbf{c}_k^{(i_k)}$ is the i_k-th codeword of codebook $\mathcal{C}_k$ and concatenate these vectors into a matrix used to compute $\mathbf{R}$, N_{cov} being the chosen number of codebooks used for the estimation. The direct approach would be to have $N_{cov} = N_{max}$, however the number of combinations to form is $1024^{N_{cov}}$ for EnCodec and AudioDec and $16^{N_{cov}}$ for Lyra V2: it would be intractable is this case. The proposed approach to estimate $\mathbf{R}$ is to use the first N_{cov} codebooks that capture most of the information on the latent distribution; for instance, one may set $N_{cov} = 2$ when $B = 10$ bits (EnCodec, AudioDec), and $N_{cov} = 5$ when $B = 4$ bits (Lyra V2). This information "ordering", i.e, the fact that the first codebooks contain most of the information, can be justified by the way RVQ is structured and trained. RVQ has a hierarchical structure, being a cascade of VQ stages. As described in [14], RVQ is learned using "quantizer dropout": for each batch of training data, the number of stages effectively used to quantize the batch of latent vectors is randomly drawn between 1 and N_{max}. Note that the covariance matrix $\mathbf{R}$ can be approximately estimated with N_{cov} up to N_{max}, assuming

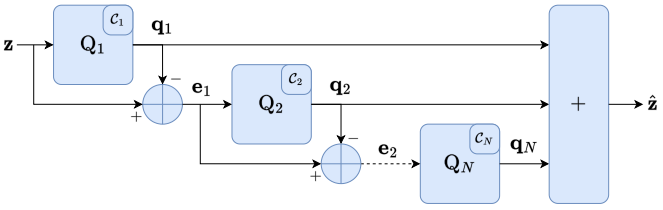

Fig. 2. RVQ: Cascade of N Vector Quantizers Q_i, each representing the residual error of the preceding stage. The reconstructed vector is the sum of decoded codewords.

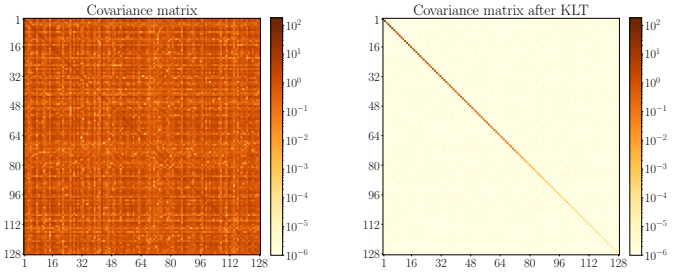

Fig. 3. Covariance matrix before and after KLT transformation.

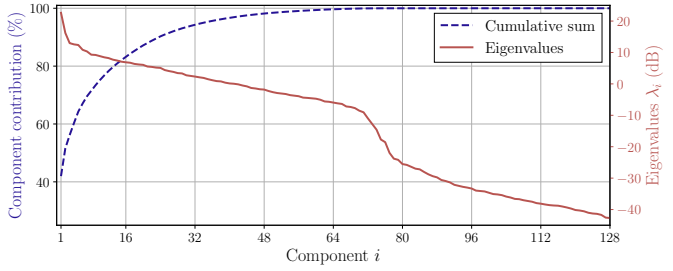

Fig. 4. Eigenvalues λ_i for EnCodec. In red: λ_i in dB. In blue: cumulative sum of λ_i, in percentage of total sum.

uncorrelated codebook stages; we verified empirically that results are not significantly affected using higher values of N_{cov}, results in Section V show that limiting N_{cov} to a value corresponding to a lower bitrate still gives satisfying results even at higher bit rates.

The matrix $\mathbf{R}$ typically shows a large amount of correlation between latent space vectors. Hence, we perform a Karhunen-Loève transform (KLT), by diagonalizing $\mathbf{R}$ as $\mathbf{R} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$ where $\mathbf{U}$ is the eigenvector matrix and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_i)$ is the diagonal matrix of eigenvalues. Decorrelated latent vectors are then obtained using the eigenvector matrix: $\mathbf{y} = \mathbf{U}^{\mathrm{T}}\mathbf{z}'$, where $\mathbf{z}'$ is the centered latent vector.

For EnCodec, the RVQ codebooks are retrieved from their publicly available pre-trained model. Fig. 3 shows the covariance matrix before and after decorrelation using the KLT (where $N_{cov} = 2$), and Fig. 4 shows eigenvalues in dB (sorted in descending order). The cumulative sum of the eigenvalues is also plotted to highlight their relative contribution. There is a substantial drop of about 20 dB between 64 and 80 dimensions. This seems to indicate that, after decorrelation, most of audio information seems to be essentially conveyed within a subspace having a dimension between 64 and 80. Results for Lyra V2 and AudioDec are presented in Section V-A.

IV. Optimization of RVQ

We describe a method designed to reduce codebook storage and computational complexity for RVQ. It leverages on the concentration of useful information on a subset of the (transformed) latent space components. Fig. 5 shows the principle of the modified RVQ. For an audio signal frame $\mathbf{x}$ of length L, the encoder $\mathcal{E}$ generates a latent vector $\mathbf{z} = \mathcal{E}(\mathbf{x}) \in \mathbb{R}^d$. The

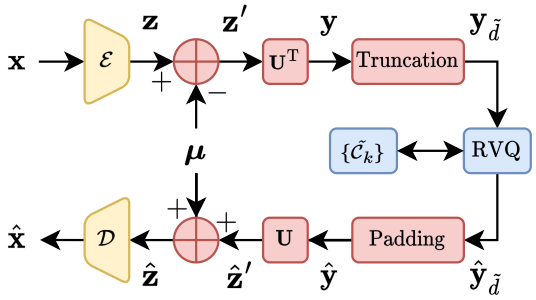

Fig. 5. Proposed method to optimize RVQ.

KLT is applied on the centered vector $\mathbf{z}' = \mathbf{z} - \boldsymbol{\mu}$ to obtain the vector $\mathbf{y} = \mathbf{U}^{\mathrm{T}}\mathbf{z}'$ with decorrelated components. This vector is then truncated to its first $\tilde{d}$ components, assuming eigenvalues are sorted in descending order (as in Fig. 4). Eq. 1 summarizes the processing, where $[\,\cdot\,]_{1:\tilde{d}}$ is the operator truncating to the first $\tilde{d}$ components:

$$\mathbf{y}_{\tilde{d}} = \left[\mathbf{U}^{\mathrm{T}}(\mathbf{z} - \boldsymbol{\mu})\right]_{1:\tilde{d}} \,. \tag{1}$$

This truncated vector is then quantized by RVQ with transformed codebooks: $\hat{\mathbf{y}}_{\tilde{d}} = \mathrm{RVQ}(\mathbf{y}_{\tilde{d}})$. The inverse process is then applied, that is, padding with $d - \tilde{d}$ zeros, inverse rotation $\hat{\mathbf{z}}' = \mathbf{U}\hat{\mathbf{y}}$ and addition of the mean $\boldsymbol{\mu}$. This is summed up by Eq. 2, where $[\,\cdot\,|\mathbf{0}_{d-\tilde{d}}]$ is the operator padding $d - \tilde{d}$ null components at the end of the vector:

$$\hat{\mathbf{z}} = \mathbf{U}\ \left[\hat{\mathbf{y}}_{\tilde{d}}|\mathbf{0}_{d-\tilde{d}}\right] + \boldsymbol{\mu}\,. \tag{2}$$

Finally, the decoder $\mathcal{D}$ generates the reconstructed audio $\hat{\mathbf{x}} = \mathcal{D}(\hat{\mathbf{z}})$.

RVQ encoding and decoding needs to be adapted for $\tilde{d}$-dimensional vectors having undergone the transformation of Eq. 1. Thus, the original codebooks $\mathcal{C}_k$ go through the same processing: KLT is applied and the codewords are truncated to $\tilde{d}$ dimensions. Since the mean $\boldsymbol{\mu}$ is subtracted to the latent vector $\mathbf{z}$, $\boldsymbol{\mu}$ has also to be is subtracted to the first codebook $\mathcal{C}_1$. The transformation to obtain new codebooks $\tilde{\mathcal{C}}_k$ is summarized in Eq. 3.

$$\begin{cases} \tilde{\mathcal{C}}_1 = \left[\mathbf{U}^{\mathrm{T}}(\mathcal{C}_1 - \boldsymbol{\mu})\right]_{1:\tilde{d}} \\ \tilde{\mathcal{C}}_k = \left[\mathbf{U}^{\mathrm{T}}\mathcal{C}_k\right]_{1:\tilde{d}} \quad k = 2, \ldots, N_{max} \end{cases} \tag{3}$$

The proposed method does not require any extra bitrate as the parameters $\boldsymbol{\mu}$ and $\mathbf{U}$ are estimated offline and do not change over time. The eigenvector matrix $\mathbf{U}$ and the mean vector $\boldsymbol{\mu}$ are pre-computed and stored, so that no additional information needs to be sent to the decoder. In practice, $\boldsymbol{\mu}$ is the mean of codebook $\mathcal{C}_1$ (to be consistent with the definition of $\tilde{\mathcal{C}}_1$ in Eq. 3) and $\mathbf{U}$ is computed using the first N_{cov} codebooks $\mathcal{C}_1$ to $\mathcal{C}_{N_{cov}}$.

V. Evaluation and Results

A. Latent space analysis (Lyra V2, AudioDec)

We report additional results for the analysis described in Section III for Lyra V2 [2], and AudioDec [16] in Fig. 6. Results for EnCodec can be found in Section III.

For AudioDec, we used the pre-trained model `libritts_sym` (24 kHz) [27] ; the number of codebooks for covariance estimation is $N_{cov} = 2$, as for EnCodec. The distribution of (sorted) eigenvalues indicates that the transformed latent space for AudioDec could be truncated to the essential dimension $\tilde{d} = 32$.

We used the available C++ implementation of Lyra V2 [2] to extract RVQ codebooks, with $N_{cov} = 5$. As shown in Fig. 6, there is no clear drop in eigenvalues, contrary to EnCodec and AudioDec. Truncation does not seem to be possible for Lyra V2 and the essential dimension has to be kept to $\tilde{d} = d = 64$.

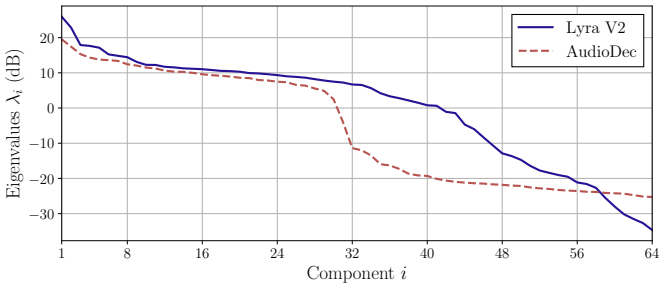

Fig. 6. Eigenvalues for Lyra V2 and AudioDec.

B. Choice of truncation (EnCodec)

To determine the truncation which leads to no audio degradation, we used the ViSQOL metric [28] in audio mode on a subset (around 25 mn) of VCTK [29] downsampled to 24 kHz. Objective tests were conducted at all EnCodec bitrates: 1.5, 3, 6, 12, and 24 kbps. The VCTK subset was created by randomly selecting 4 audio files for each of the 106 talkers in VCTK. We verified that the distribution of audio levels in this subset followed the distribution of the entire VCTK dataset.

Fig. 7 shows the average ViSQOL score with truncations from $\tilde{d} = 40$ to 80 (with a step of 8); the original EnCodec (no truncation) corresponds to $\tilde{d} = d = 128$. The 95% confidence interval was computed to check for any statistically significant difference. For a truncation to $\tilde{d} = 80$ or 72, objective audio quality is the same as original EnCodec, whereas there is a clear degradation of the ViSQOL score when $\tilde{d} \leq 64$. Based on this preliminary analysis, we decided to conduct subjective tests to verify whether there is any effective audio degradation when truncating latent space to $\tilde{d} = 64$ or 72 dimensions.

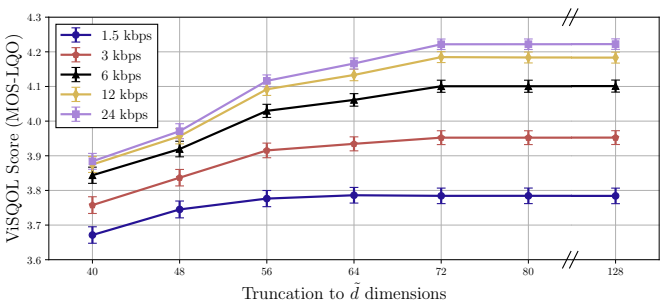

Fig. 7. ViSQOL score (audio mode) for several truncations and bitrates with EnCodec.

C. Audio quality validation for optimized RVQ (EnCodec)

We verify that the proposed RVQ optimization method described in Section IV does not degrade quality after truncation of the transformed latent space. For this part EnCodec was used with respectively $\tilde{d} = 64$ or 72 (modified) and $\tilde{d} = d = 128$ (original). In other words, we compared decoded signals when RVQ is kept unchanged or modified according to Fig. 5.

AB tests (with reference) were conducted at all EnCodec bitrates: 1.5, 3, 6, 12, and 24 kbps. In each test, 10 "critical" items (5 speech items + 5 music and mixed content items) were preselected by searching worst-case items identified by objective tests in two separate sets: VCTK subset described in Section V-B and an internal music and mixed content database obtained by normalizing 30 audio files at three audio levels: -36, -26 and -16 dB LKFS [30]. The worst-case items were identified using ViSQOL in audio mode. The difference score was computed for each item as $\Delta_S = [S(\mathbf{x}, \hat{\mathbf{x}}_{mod}) - S(\mathbf{x}, \hat{\mathbf{x}}_{ori})]$ where $S(\cdot, \cdot)$ is the score associated to the metric (ViSQOL), $\mathbf{x}$ is the uncoded audio, $\hat{\mathbf{x}}_{ori}$ and $\hat{\mathbf{x}}_{mod}$ is the audio reconstructed by the original and modified EnCodec, respectively.

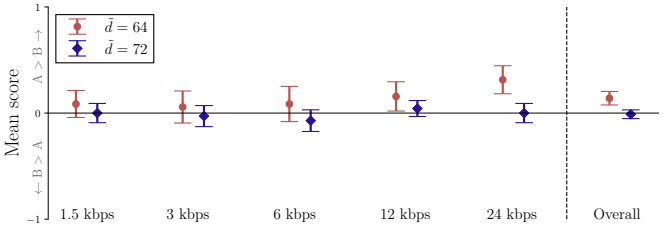

Fig. 8. AB test results: A = original EnCodec, B=modified EnCodec ($\tilde{d} = 64$ or 72). Average score (over 10 items) is shown (with 95% confidence interval).

In total, 7 expert listeners conducted two sets of 5 subjective AB tests – one set per truncation ($\tilde{d} = 64$ and 72), one test (comprising 10 critical items) per EnCodec bitrate. A 5-point scale (from $+2$ to -2) was used for grading: $0 \rightarrow$ A and B are the same, $\pm 1 \rightarrow$ A (resp. B) is better than B (resp. A), $\pm 2 \rightarrow$ A (resp. B) is much better than B (resp. A). Test results are presented in Fig. 8; when the overall score is above 0, A (original EnCodec) is better, and when it is negative, B (modified EnCodec) is better. Overall there were no noticeable differences or artifacts for modified EnCodec with $\tilde{d} = 72$; the same observation applies to average scores over individual critical items (not shown in Fig. 8) for $\tilde{d} = 72$, scores were sometimes in favor of A and sometimes in favor of B, therefore testers could not differentiate between the two versions of EnCodec. However, for $\tilde{d} = 64$, we can observe that original EnCodec is significantly better at higher bitrates (12 and 24 kbit/s); truncating at $\tilde{d} = 64$ brings some audio degradation that is noticeable, which is not the case when truncating at $\tilde{d} = 72$. This strengthen the observation made with Fig. 7 to choose the dimension for truncation. Results also indicate that estimating the covariance $\mathbf{R}$ using only the first RVQ codebooks may be suboptimal for high bit rates;

still, when $\tilde{d}$ is sufficiently high, this shortcoming in covariance estimation has no penalty.

D. Complexity and storage gain for optimized RVQ (EnCodec)

The RVQ optimization method described in Section IV can reduce VQ (nearest neighbor search) complexity and codebook storage.

For EnCodec, $N_{max} = 32$ codebooks $\mathcal{C}_k$ are represented with $32 \times 128 \times 1024$ floats. The truncation to $\tilde{d} = 72$ dimensions instead of $d = 128$ reduces this number to $32 \times 72 \times 1024$. Note that the mean $\boldsymbol{\mu}$ and the eigenvectors matrix $\mathbf{U}$ require storing $128 + 128 \times 128$ additional floats. Overall, the gain in storage for EnCodec is 43.4% when truncating to $\tilde{d} = 72$. When considering the entire neural codec, the encoder and decoder consist of about 14.85M parameters (according to the torchinfo library) while quantization codebooks are composed of about 4.2M parameters. Thus, this approach reduces the total model storage by 10%.

For computational complexity, two additions ($+\boldsymbol{\mu}$ and $-\boldsymbol{\mu}$) and two matrix multiplication ($\mathbf{U}$ and $\mathbf{U}^{\mathrm{T}}$) are required, together with truncation and padding operations. On the other hand, nearest neighbor search in VQ encoding step is performed on vectors of dimension $\tilde{d} = 72$ instead of $d = 128$, resulting in a gain in computational complexity for the quantization part. To be more accurate, we estimated the number of floating-point operations needed in both cases (original and modified EnCodec) considering comparisons, additions and MAC (multiply-accumulate) operations. For a codebook with n codewords of dimension δ, the nearest neighbor search for one codebook is estimated by $2 \times \delta \times n + (n - 1)$ and the overhead added for the addition of means and the matrix multiplications corresponds to $2 \times (\delta + \delta \times \delta)$ operations. By replacing δ with d and $\tilde{d}$ for the original and modified EnCodec, respectively, the gain in complexity reaches 43.2% when using all the codebooks (bitrate of 24 kbps), with a drop to 37.3% when using two codebooks (bitrate of 1.5 kbps).

E. Verification of backward compatibility (EnCodec)

Objective and subjective test results reported in Sections V-B and V-C evaluate symmetric cases: original encoder–original decoder vs. modified encoder–modified decoder. One key feature of the proposed method is to guarantee backward compatibility, that is, the modified encoder and decoder are compatible with their original counterparts. To verify this, we conducted extra objective tests to evaluate cross-cases, where the test dataset and metric are the same as in Section V-B and truncation is $\tilde{d} = 72$.

The results showed differences between the original EnCodec and the cross-cases to be zero up to the third decimal, with a maximum difference of 8.10^{-2} MOS-LQO among all the tested items and at all bitrates. This validates that modifications according to the proposed method are backward compatible.

VI. Conclusion

In this work we used the KLT on learned codebooks to decorrelate and reduce the dimension of latent vectors in a post-training phase. This method can bring a significant reduction in quantization complexity and codebook storage, depending on the codec. Experimental results show that a gain of about 43.4% can be obtained for EnCodec, with no significant quality degradation. In future work we plan to compare RVQ with alternative methods, including KLT-based quantization derived from the proposed latent space analysis.

References

[1] Google, "Lyra, v0.0.2," https://github.com/google/lyra/releases/tag/v0.0.2.
[2] ——, "Lyra, v1.3.2," https://github.com/google/lyra/releases/tag/v1.3.2.
[3] R. Vipperla *et al.*, "Bunched LPCNet : Vocoder for Low-cost Neural Text-To-Speech Systems," arXiv:2008.04574, 2020.
[4] Z. Borsos *et al.*, "AudioLM: a Language Modeling Approach to Audio Generation," arXiv:2209.03143, 2023.
[5] J. Copet *et al.*, "Simple and controllable music generation," arXiv:2306.05284, 2024.
[6] A. van den Oord *et al.*, "WaveNet: A Generative Model for Raw Audio," in *arXiv:1609.03499*, 2016.
[7] S. Mehri *et al.*, "SampleRNN: An unconditional end-to-end neural audio generation model," in *Proc. ICLR*, 2017.
[8] N. Kalchbrenner *et al.*, "Efficient neural audio synthesis," in *Proc. ICML*, 2018, pp. 2410–2419.
[9] W. B. Kleijn *et al.*, "Wavenet based low rate speech coding," in *Proc. ICASSP*, 2018.
[10] J.-M. Valin and J. Skoglund, "LPCNet: Improving Neural Speech Synthesis through Linear Prediction," in *Proc. ICASSP*, 2019.
[11] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural Discrete Representation Learning," in *Proc. NeurIPS*, 2018.
[12] K. Kumar *et al.*, "MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis," in *Proc. NeurIPS*, 2019.
[13] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis," arXiv:2010.05646, 2020.
[14] N. Zeghidour *et al.*, "SoundStream: An End-to-End Neural Audio Codec," *IEEE/ACM Trans. TASLP*, 2021.
[15] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High Fidelity Neural Audio Compression," in *arXiv:2210.13438*, 2022.
[16] Y.-C. Wu *et al.*, "Audiodec: An open-source streaming high-fidelity neural audio codec," in *Proc. ICASSP*, 2023, pp. 1–5.
[17] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1991.
[18] 3GPP TR 26.952, "Codec for Enhanced Voice Services (EVS); Performance characterization."
[19] N. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Prentice-Hall, 1984.
[20] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of educational psychology*, vol. 24, 1933.
[21] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, 2008.
[22] L. Bonheme and M. Grzes, "FONDUE: An algorithm to find the optimal dimensionality of the latent representations of variational autoencoders," arXiv:2209.12806, 2022.
[23] A. Caillon and P. Esling, "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis," *CoRR*, vol. abs/2111.05011, 2021.
[24] C.-H. Pham and al., "PCA-AE: Principal Component Analysis Autoencoder for Organising the Latent Space of Generative Networks," *Journal of Mathematical Imaging and Vision*, vol. 64, no. 5, Jun. 2022.
[25] M. Ho, X. Zhao, and B. Wandelt, "Information-ordered bottlenecks for adaptive semantic compression," 2023.
[26] R. Kumar and al., "High-Fidelity Audio Compression with Improved RVQGAN," arXiv:2306.06546, 2023.
[27] Meta, "AudioDec," https://github.com/facebookresearch/AudioDec.
[28] M. Chinen *et al.*, "ViSQOL v3: An Open Source Production Ready Objective Speech and Audio Metric," in *Proc. QoMEX*, 2020.
[29] J. Yamagishi *et al.*, "CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit," 2019.
[30] ITU-R BS.1770–4, "Algorithms to measure audio programme loudness and true-peak audio level," Oct. 2015.