Spherical Lattice Vector Quantization in Neural Audio Coding

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

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

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

***Abstract*—In neural audio coding, latent space quantization is often trained together with the rest of the model. In this work, we investigate the use of algebraic vector quantization (VQ) in a shape-gain approach, to modify the residual vector quantization (RVQ) method in the DAC neural audio codec. Results on speech signals show that the overall performance of DAC with the proposed modifications is nearly equivalent to the original DAC model, while offering significant advantages: virtually no codebook storage, fast nearest neighbor search and indexing, codebook training limited to one gain optimization per quantization stage.**

***Index Terms*—neural audio coding, lattice, spherical vector quantization, audio quality.**

I. Introduction

In recent years, the field of audio coding has been shaken up by deep learning technologies. The use of neural networks enables much lower data rates in a variety of applications, such as voice/video calling, music storage. More recently, neural audio coding has been essential to "tokenize" audio to interface with large language models, for instance in speech-to-speech translation [1] or music generation applications [2]. Autoencoder architectures with GAN-type training (using discriminators) have shown great results, with codecs such as SoundStream [3], EnCodec [4] or Descript Audio Codec (DAC) [5]. Other approaches such as diffusion models are also being investigated [6], [7].

One fundamental building block of neural audio coding is the quantization of latent space. Residual Vector Quantization (RVQ) [3] – which is revisiting the well-known multi-stage VQ [8] – is a popular method enabling multi-rate operation. Alternative methods have been proposed, such as Finite Scalar Quantization [9] or Lookup Free Quantization [10]. Many quantization techniques have been available in the literature for many years. Among them, algebraic vector quantization, in particular spherical lattice VQ [11], has been successfully applied to "traditional" speech and audio coding – for instance the method in [12], leveraging spherical lattice VQ, has been adapted in several "traditional" codec standards (e.g., AMR-WB+, USAC, EVS, IVAS).

In this work, we propose to study the use of such algebraic VQ within the state-of-the-art neural audio codec Descript Audio Codec (DAC) [5]. Latent space quantization in DAC is based on RVQ, however it differs from the RVQ method in SoundStream and EnCodec by the use of projection and expansion layers to perform code lookup in a space with lower dimension than the latent space.

The main contributions of this paper are listed below:

- We integrate lattice VQ within DAC, with shape-gain approach [13], considering two cases: without DAC retraining vs. with a complete model retraining;
- We present fast lattice quantization algorithms in dimension 8 with a performance validation on the Gaussian source – nearest neighbor search is derived from [14], indexing and decoding algorithms are new, building on top of [14].
- We report the (objective and subjective) speech quality of the proposed modifications to DAC.

II. Brief Review of Descript Audio Codec (DAC)

The DAC codec [5] belongs to the family of autoencoders trained in a GAN framework. At inference time, it comprises a convolutional encoder and decoder, and latent space quantization. Several variants of the model are proposed in [5]; we consider only the full-band variant, operating with mono audio sampled at 44.1kHz. Each input frame of 512 samples is converted by the encoder into a 1024-dimensional latent vector; after quantization, the quantized latent vector is converted back into audio by the decoder.

Multi-stage quantization, known as RVQ [3] in neural audio coding, is used to operate at several bitrates. Quantization stages Q_k are cascaded, as shown in Fig. 1. The number of quantization stages is flexible, ranging from 1 to K_{max}. In DAC, $K_{max} = 9$ and each stage has a codebook of $M = 1024$ codewords (10 bits). The search for the best codeword is carried out in a lower dimensional space than latent space. Linear projection and expansion layers (P_k and E_k, respectively) are added in DAC to go from the latent space dimension 1024 to the lookup dimension 8.

We analyzed the 8-dimensional pretrained RVQ codebooks in the original DAC model by inspecting histograms of codeword norms (L2) in each RVQ stage, as shown in Fig. 2. The distribution is "narrow" for the last 6 RVQ stages, which may indicate that learned codewords are approximately spherical. In addition, we calculated the minimum angular distance between codewords in each stage, to study the potential uniform

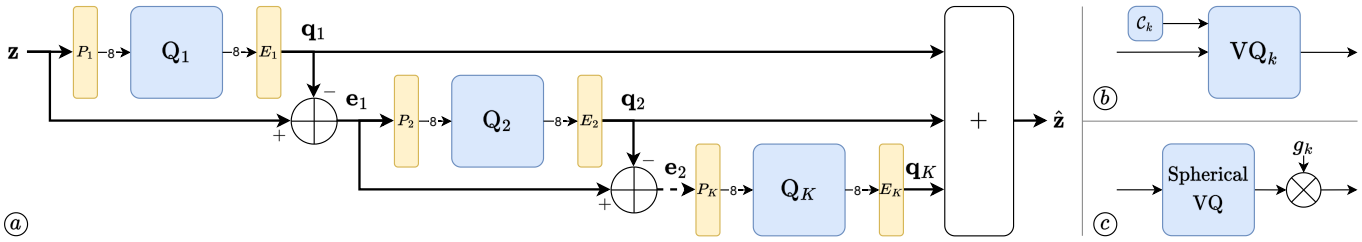

Fig. 1: Latent space quantization in DAC: (a) Cascade of K_{max} stages comprising projection (P_k), vector quantization (Q_k), and expansion (E_k) blocks; (b) Q_k in DAC: stochastic VQ using learned codebooks $\mathcal{C}_k$; (c) Proposed replacement for Q_k: shape-gain VQ based on spherical lattice VQ.

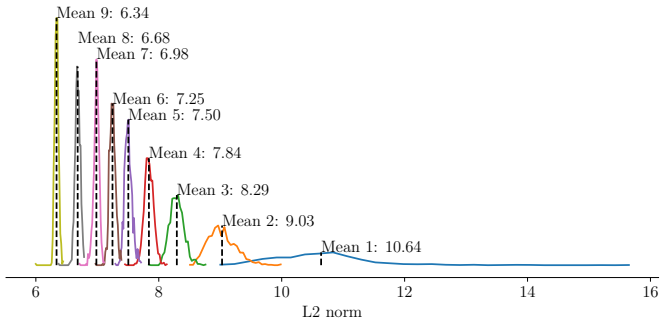

Fig. 2: Histograms of codeword (L2) norms in the $K_{max} = 9$ pre-trained RVQ codebooks of DAC. Each label 'Mean k' indicates the mean norm of codewords in $\mathcal{C}_k$.

spherical distribution of codewords. The minimum distance is around 35.5 deg. in the first stage, and it is around 38 deg. in other stages – this can be compared to the expected distance of 44 deg. for uniformly distributed codes in dimension 8 [15]. These observations are not totally surprising, given that the cosine similarity criterion is used for codebook search in DAC. This insight on DAC codebooks motivated the use of spherical vector quantization in this work.

III. Spherical Lattice Vector Quantization

A. Generalities on spherical lattice quantization

Spherical VQ consists in representing a source $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ by a codebook $\mathcal{C} = \{\mathbf{y}_0, \ldots, \mathbf{y}_{M-1}\} \in \mathcal{S}_{n-1}^M$ which is a finite subset of $(n-1)$-sphere $\mathcal{S}_{n-1}$ of unit radius (with no loss of generality), where

$$\mathcal{S}_n = \left\{ \mathbf{x} \in \mathbb{R}^{n+1} \;\middle|\; ||\mathbf{x}||^2 = x_1^2 + \cdots + x_{n+1}^2 = 1 \right\} \quad (1)$$

When the distortion is defined as the (mean) squared error, it is easy to show that the nearest neighbor search of $\mathbf{x}$ in $\mathcal{C}$ boils down to *maximizing* the dot product $\mathbf{x}^T \cdot \mathbf{y}_i$ over $i = 0, ..., M-1$, which is equivalent to the so-called *cosine similarity* (up to a fixed scale).

Spherical codes play an important role in source coding due to the "sphere hardening" property of the Gaussian source [16], i.e. the n-dimensional vectors of an i.i.d Gaussian source fall with high probability on a spherical surface when $n \to \infty$. A comprehensive review of spherical codes can be found in [17].

In this work we study the application of a particular type of spherical code where the codebook is derived from a lattice. A (regular) lattice Λ in $\mathbb{R}^n$ is a set of discrete points defined by $\Lambda = \mathbf{k}.\mathbf{G}$, where $\mathbf{k}$ is an integer vector in $\mathbb{Z}^n$ and $\mathbf{G}$ is a generator matrix obtained by stacking a set of linear independent basis vectors. A trivial lattice example is $\mathbb{Z}^n$ for which the identity matrix is a generator matrix. A comprehensive treatment of lattices can be found in [18].

Spherical lattice VQ has been introduced in [11], and the special cases of spherical VQ using the Gosset and Leech lattices in dimension 8 and 24 has been studied in [14] and [19], respectively.

B. Spherical lattice quantization based on the Gosset lattice

Since RVQ in DAC operates with codebooks in dimension 8, we focus here only on the case of the Gosset lattice denoted RE_8. We recall the definition of $RE_8 = 2D_8^+$ [18] :

$$RE_8 = 2D_8 \cup \{2D_8 + (1, \ldots, 1)\}, \quad (2)$$

with

$$D_8 = \left\{ (x_1, \ldots, x_8) \in \mathbb{Z}^8 \;\middle|\; \sum_{i=1}^{8} x_i \text{ is even} \right\}. \quad (3)$$

A point $\mathbf{x} = (x_1, \ldots, x_8)$ in RE_8 verifies the following properties:

- All elements x_i are integers
- The sum of all elements x_i is a multiple of 4
- All elements x_i have the same parity, i.e. they are either all even or all odd
- The sum of x_i^2 is a multiple of 8

The latter property can be interpreted geometrically by decomposing RE_8 into an union of concentric spherical shells (or orbits) of index m centered on the origin and of radius $2\sqrt{2m}$, where $m \geq 0$ is an integer; the set of lattice points on a given shell may be used to define a spherical code [14].

A special property of RE_8 is that any permutation of a point $\mathbf{x}$ in a RE_8 shell remains in the same shell. One can therefore decompose each lattice shell into a union of permutation codes, where one vector called "leader" generates a subset of equivalent vectors on the same shell by permutation [14]. A "leader" is an integer vector whose elements are sorted in

descending order, this vector may be either a "signed leader" or an "absolute leader". Codewords related to a signed leader are generated through permutations only, while codewords related to an absolute leader are generated by permutation and sign changes, with the additional constraint in RE_8 that the parity of the total number of negative elements is identical to the predefined parity of the absolute leader (denoted ISIG in [14]). Fast nearest neighbor search algorithms can be designed in a spherical RE_8 codebook by exploiting this concept of "leaders". The full search is replaced by a search limited to a small set of leaders.

C. Quantization algorithms based on absolute leaders

In the following, given a vector $\mathbf{x} = (x_1, \ldots, x_8)$, the notation $|\mathbf{x}|$ indicates that the absolute value is applied to $\mathbf{x}$ element-wise, such that $|\mathbf{x}| = (|x_1|, \ldots, |x_8|)$. Moreover, sorting elements of $\mathbf{x}$ in descending order results in a new vector denoted $\tilde{\mathbf{x}}$. With these conventions, an absolute leader will be noted $|\tilde{\mathbf{y}}_\mathbf{k}|$ because its elements are by definition all positive and sorted in descending order.

In this work we use a fast codebook search based on (normalized) absolute leaders in RE_8. The nearest neighbor search algorithm of a given input $\mathbf{x} = (x_1, \ldots, x_8)$ in a spherical codebook defined by a list of L absolute leaders $\{|\tilde{\mathbf{y}}_1|, \ldots, |\tilde{\mathbf{y}}_L|\}$, with associated sign parity ISIG_k, $k = 1, \ldots, L$, is detailed below.

1) Compute the *absolute vector* $|\tilde{\mathbf{x}}|$ by permuting elements of $|\mathbf{x}|$ to sort them in descending order. Save this permutation.
2) Compute sign parity $\Pi(\mathbf{x})$ of $\mathbf{x}$:

$$\Pi(\mathbf{x}) = \text{parity}\left(\sum_{j=1}^{8} \text{sign}(x_j)\right), \tag{4}$$

   where $\text{parity}(s) = 0$ if s is even and 1 if s is odd; $\text{sign}(x) = 1$ if $x < 0$, 1 otherwise.
3) Compute dot product with L normalized leaders with a penalty term accounting for sign parity mismatch:

$$d(k) = |\tilde{\mathbf{x}}|^T \cdot \frac{|\tilde{\mathbf{y}}_k|}{\parallel |\tilde{\mathbf{y}}_k| \parallel} - 2\epsilon, \qquad k = 1, \ldots, L \tag{5}$$

   where $\epsilon = 0$ if $\Pi(\mathbf{x}) = \text{ISIG}_k$ and $|\tilde{x}_8| . \frac{|\tilde{y}_{k,8}|}{\parallel |\tilde{\mathbf{y}}_k| \parallel}$ otherwise.
4) Find best leader by maximizing the dot product:

$$k^* = \underset{k=1,\ldots,L}{\arg\max}\, d(k) \tag{6}$$

5) Reconstruct codeword $\mathbf{y} = (y_1, \ldots, y_8)$ as follows:
   a) Initialize from absolute leader taking into account sign parity mismatch: $y_j = |\tilde{y}_{k^*,j}| / \parallel |\tilde{\mathbf{y}}_{k^*}| \parallel$, $j = 1, \ldots, 8$ and negate last element y_8 if $\Pi(\mathbf{x}) \neq \text{ISIG}_{k^*}$
   b) Apply inverse permutation (from Step 1) to $\mathbf{y}$
   c) Apply signs from $\mathbf{x}$: negate y_j if $x_j < 0$, $j = 1, \ldots, 8$

The above algorithm is essentially the same as in [14], the main difference is that we allow absolute leaders from different shells to be in the spherical codebook, normalizing the dot product by $\parallel |\tilde{\mathbf{y}}_k| \parallel$.

Spherical codebook indexing based on absolute leaders is not described in [14]. We propose the following indexing algorithm. The index i of the codeword $\mathbf{y}$ is defined as:

$$i = \sigma(\mathbf{y})\nu(|\tilde{\mathbf{y}}_{\mathbf{k}^*}|) + \rho(|\mathbf{y}|) + \sum_{k=1}^{k^*-1} \mathcal{N}_k \tag{7}$$

where $\sigma(\mathbf{y})$ is the sign code of $\mathbf{y}$, $\nu(|\tilde{\mathbf{y}}|)$ is the number of permutations of the absolute leader $|\tilde{\mathbf{y}}_{\mathbf{k}^*}|$, $\rho(|\mathbf{y}|)$ is the rank of permutation of $|\mathbf{y}|$ (given that $|\mathbf{y}|$ is a permutation of $|\tilde{\mathbf{y}}_{\mathbf{k}^*}|$), and $\mathcal{N}_k$ is the total number of possible signed permutations of $|\tilde{\mathbf{y}}_{\mathbf{k}^*}|$ in RE_8. The sign code $\sigma(\mathbf{y})$ of $\mathbf{y}$ depends on whether the absolute leader $|\tilde{\mathbf{y}}_{\mathbf{k}^*}|$ is even or odd. For an even leader, $\sigma(\mathbf{y})$ is obtained by stacking sign bits of non-zero elements of $\mathbf{y}$. For an odd leader, the first 7 signs of $y_1 ... y_7$ are stacked. In this work we simply use Schalkwijk's formula [20] to compute the rank of permutation $\rho(|\mathbf{y}|)$.

The decoding of an index i (obtained according to Eq. 7) is summarized below, using the following algorithm:

1) Find absolute leader index k^* by comparing successively the value of i against a precomputed table giving $\sum_{k=1}^{k*-1} \mathcal{N}_k$ for $k^* = 1, \ldots, L$.
2) Update i by subtracting the corresponding cardinality offset: $i \leftarrow i - \sum_{k=1}^{k^*-1} \mathcal{N}_k$
3) Get $\sigma(\mathbf{y})$ and $\rho(|\mathbf{y}|)$ as the quotient and remainder (respectively) in the integer division of i by $\nu(|\tilde{\mathbf{y}}_{\mathbf{k}^*}|)$
4) Find $\mathbf{y}$ by permuting the absolute leader $|\tilde{\mathbf{y}}_{\mathbf{k}^*}|$ based on rank of permutation $\rho(|\mathbf{y}|)$
5) Negate elements of $\mathbf{y}$ based on the sign code $\sigma(\mathbf{y})$

D. Evaluation on the Gaussian source

Table I defines a set of RE_8 spherical codebooks at $R = 8$, 10 and 12 bits per vector. At 8 bits, the codebook consists of the shell 1 (240 points) and an extra leader (16 points) from shell 2 to $M = 256$ codewords. At 10 bits, we define two possible variants with $M = 1024$ codewords: one with a single absolute leader (incomplete shell 2), an alternative (denoted 'alt'.) with 4 absolute leaders (mixing shells 1, 3, 5 and 10). At 12 bits, the codebook of $M = 4080$ codewords consists of shell 2 (2160 points) and one leader from shell 1 (128 points) and shell 3 (1792 points). A Monte Carlo simulation has been conducted using trials of 100,000 random vectors $\mathbf{x}$ with a zero-mean unit-variance Gaussian source in dimension 8. Each input vector $\mathbf{x}$ is quantized in $\mathbf{y}$ using the RE_8 spherical quantization at R bits (based on Table I) with search, indexing and decoding algorithms described in the previous section. A fixed scaling factor g_{opt} is applied to each codeword $\mathbf{y}$ determined (optimized) empirically to minimize the mean squared error $\mathbb{E}[||\mathbf{x} - g_{opt}\mathbf{y}||^2]$. Table II reports the resulting signal to noise ratio (SNR), compared to the rate-distortion (R-D) bound given by SNR=6.02 $(R/8)$ dB. For the two codebook variants at 10 bits, we observe that the codebook with a single absolute leader ($[3, 1, \ldots, 1]$) gives a better SNR.

IV. Experimental results on DAC

To allow for a fair comparison, we replaced the 10-bit RVQ codebooks from DAC (Fig. 1 (b)) by shape-gain codebooks

TABLE I: RE_8 spherical codebooks.

R	k	Absolute leader $\lvert\tilde{\mathbf{y}}_k\rvert$	ISIG_k	$\mathcal{N}_k$
8	0	[2, 2, 0, 0, 0, 0, 0, 0]	0	112
	1	[1, 1, 1, 1, 1, 1, 1, 1]	0	128
	2	[4, 0, 0, 0, 0, 0, 0, 0]	0	16
10	0	[3, 1, 1, 1, 1, 1, 1, 1]	1	1024
10 (alt.)	0	[1, 1, 1, 1, 1, 1, 1, 1]	0	128
	1	[6, 2, 0, 0, 0, 0, 0, 0]	0	224
	2	[4, 4, 4, 0, 0, 0, 0, 0]	0	448
	3	[8, 4, 0, 0, 0, 0, 0, 0]	0	224
12	0	[1, 1, 1, 1, 1, 1, 1, 1]	0	128
	1	[4, 0, 0, 0, 0, 0, 0, 0]	0	16
	2	[2, 2, 2, 2, 0, 0, 0, 0]	0	1120
	3	[3, 1, 1, 1, 1, 1, 1, 1]	1	1024
	4	[2, 2, 2, 2, 2, 2, 0, 0]	0	1792

TABLE II: Performance of RE_8 spherical vector quantization.

R	g_{opt}	SNR (dB)	R-D bound (dB)
8	2.29	4.96	6.02
10	2.45	6.06	7.52
10 (alt.)	2.40	5.90	7.52
12	2.51	7.24	9.03

(Fig. 1 (c)) where the shape codebook is the 10-bit RE_8 spherical codebook defined in Table I with a single absolute leader – this case gave better performance on the Gaussian source compared to the alternative 10-bit codebook. All RE_8 quantization algorithms (nearest neighbor search, indexing and decoding) have been vectorized for use in the PyTorch framework. In each quantization stage, the RE_8 codebook is scaled by a single scale factor (gain) that is to be optimized empirically (at inference phase) or learned (at training phase).

A. Experiment 1 without retraining of DAC

We replaced DAC quantizers with spherical VQ. We tested different combinations of RVQ and spherical VQ, where the first κ quantization stages are kept as pretrained RVQ from DAC and the last $K_{max} - \kappa$ stages are replaced by shape-gain RE_8 VQ, with $\kappa \in [0, K_{max} - 1]$.

A gain g_k is defined for RE_8 lattice VQ, as shown in Fig. 1 (c). The value of g_k has been optimized heuristically based on the value $g_{opt} = 2.45$ obtained for coding the 8-dimensional Gaussian source at 10 bits, with a correction factor given by the ratio between the average codeword norm in pre-trained $\mathcal{C}_k$ (i.e., 'Mean k' in Fig. 2) and the average norm of the 8-dimensional Gaussian source.

The modified DAC model where only latent space quantization is modified at inference time was evaluated using a speech test database. Following [21], we used the POLQA (P.863) objective model [22].The test dataset consisted of 36 phonetically balanced double sentences in French (6 talkers with 3 women and 3 men, 6 sentences per talker). These sentences are adapted in length and content for use with POLQA. Objective speech quality results are shown in Fig. 3. Let alone the extreme case, where all RVQ stages are replaced by lattice VQ ($\kappa = 0$), objective quality results are equivalent in all cases. It is expected that the case $\kappa = 0$ would give lower results, since the RVQ codebook in the first stage is less spherical (see Fig. 2), so spherical lattice VQ is less suitable in the first stage.

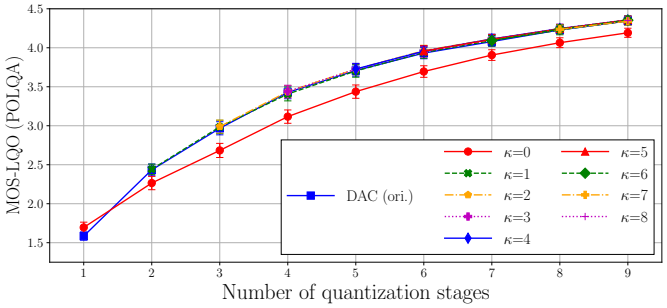

Fig. 3: Speech quality (estimated by POLQA) versus bitrate, for different configurations (no retraining of DAC).

For Experiment 1 without DAC retraining, keeping at least one stage of learned VQ seems necessary. Nevertheless, these results suggest that it is possible to replace the other 8 learned codebooks by lattice codebooks without degrading audio quality.

B. Experiment 2 with retraining of full DAC model

Results from Experiment 1 demonstrated that spherical VQ may be integrated within a pretrained version of the DAC model. The rest of the work focused on retraining the DAC model where RVQ is replaced by lattice VQ during training. We tested several cases: lattice VQ only ($\kappa = 0$), a combination of first $\kappa = 1$ and $\kappa = 2$ RVQ stage(s) following by lattice VQ stages. Here, the gains g_k in each stage using lattice VQ were initialized to 1 and may or may not be trainable as extra model parameters.

The DAC variants were trained using the original DAC source code [23] and under the same conditions except for the dataset. For the sake of reproducibility, we used the publicly available EARS speech database [24] comprising 100 hours of full-band anechoic clean speech with 107 speakers and covering different speaking styles (reading, emotional speech, conversational freeform speech, etc.). The original DAC model was also retrained on the EARS dataset for comparison purposes.

Results for retrained models are shown in Fig. 4. The same speech test dataset as Experiment 1 was used. Only the curves for a trainable gain g_k are shown, as models with a not trainable gain have poorer results. Variants with $\kappa = 0, 1,$ and 2 and the DAC model trained on EARS are very close in performance, and are at least as good as the (pre-trained) original DAC model. The original DAC deviates from these models at lower bitrates, possibly due to the more diverse audio content used in training, which could make the original DAC better for various types of audio, and a bit worse for speech on the EARS dataset compared to models trained only on speech with the EARS dataset. These results show that the overall performance of DAC with a modified latent space quantization method is nearly equivalent to the original DAC model, while offering significant advantages: virtually no codebook storage, fast nearest neighbor search and

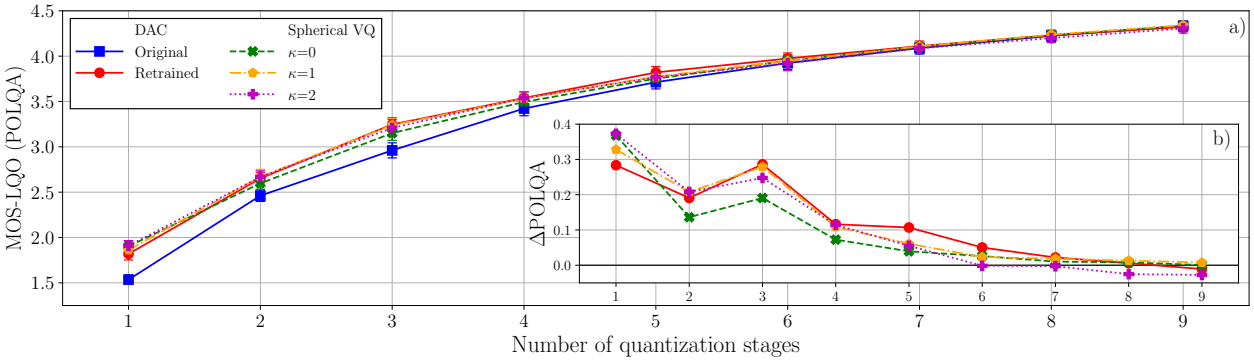

Fig. 4: Main figure (a): Speech quality (estimated by POLQA) vs. bitrate for different configurations when retraining DAC. Subfigure (b): Difference scores (ΔPOLQA) between the four retrained models and the original DAC model – a positive difference indicates an improvement over the original DAC model.

indexing, codebook training limited to one gain optimization per quantization stage.

An informal subjective RefAB test was conducted with 5 experts, comparing the retrained DAC model and proposed model with lattice VQ in all stages ($\kappa = 0$) on the 36 speech test items. As audio quality is close to transparency when using $K_{max} = 9$ quantization stages, the RefAB test was performed at a bitrate corresponding to 3 quantization stages. At this bitrate, the audio quality is still good but there are some artefacts in the two tested systems. Subjects confirmed that the quality of the two systems is equivalent: there were no significant statistical differences (for all items and overall).

C. Discussion on complexity and storage requirements

The computational complexity of RVQ is given by codebook search at encoding (estimated to approx. $25,600$ operations per codebook stage) and table look-up at decoding (negligible). For RE_8 spherical VQ at 10 bits (with a single leader), this complexity is reduced to approx. 100-200 operations per stage; search is limited to sorting the absolute vector, applying signs and inverting the permutation; indexing and deindexing may also be dramatically simplified as this boils down to indexing the signed permutations of $[3, 1, \ldots, 1]$. The $K_{max} = 9$ RVQ codebooks C_k require storing $1024 \times 8 \times 9 = 73728$ values; the RE_8 spherical VQ at 10 bits is based on a single leader – the search and indexing algorithms can be easily hard-coded for this case, and there is virtually no codebook storage required, only 9 gains g_k have to be stored.

V. Conclusion

In this work, we proposed to replace RVQ in the DAC neural audio codec by spherical algebraic VQ. The study shows that the performance obtained with spherical lattice VQ is very promising. In future work we plan to test other codebook rates than 10 bits per RVQ stage.

Acknowledgment

The authors thank Pierrick Philippe (Orange Research in Rennes, France) for discussions and proofreading.

References

[1] T. Labiausse, L. Mazaré, E. Grave, P. Pérez, A. Défossez, and N. Zeghidour, "High-fidelity simultaneous speech-to-speech translation," 2025.
[2] J. Copet *et al.*, "Simple and controllable music generation," in *Proc. NeurIPS*, 2023.
[3] N. Zeghidour *et al.*, "SoundStream: An End-to-End Neural Audio Codec," *IEEE/ACM Trans. TASLP*, 2021.
[4] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," *Transactions on Machine Learning Research*, 2023.
[5] R. Kumar *et al.*, "High-fidelity audio compression with improved rvqgan," in *Proc. NeurIPS*, 2023, pp. 27 980–27 993.
[6] Y.-C. Wu *et al.*, "Scoredec: A phase-preserving high-fidelity audio codec with a generalized score-based diffusion post-filter," in *Proc. ICASSP*, 2024.
[7] S. Welker *et al.*, "Flowdec: A flow-based full-band general audio codec with high perceptual quality," in *Proc. ICLR*, 2025.
[8] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1991.
[9] F. Mentzer, D. Minnen, E. Agustsson, and M. Tschannen, "Finite scalar quantization: VQ-VAE made simple," in *Proc. ICLR*, 2024.
[10] L. Yu *et al.*, "Language model beats diffusion - tokenizer is key to visual generation," in *Proc. ICLR*, 2024.
[11] J.-P. Adoul, "La quantification vectorielle des signaux: approche algébrique," in *Ann. des télécom.*, vol. 41, no. 3, 1986, pp. 158–177.
[12] S. Ragot *et al.*, "Low-complexity multi-rate lattice vector quantization with application to wideband TCX speech coding at 32 kbit/s," in *Proc. ICASSP*, 2004.
[13] M. Sabin and R. Gray, "Product code vector quantizers for waveform and voice coding," *IEEE Trans. AASP*, vol. 32, no. 3, pp. 474–488, 1984.
[14] C. Lamblin and J.-P. Adoul, "Algorithme de quantification vectorielle sphérique à partir du réseau de Gosset d'ordre 8," pp. 172–186, 1988.
[15] H. Cohn, "Table of spherical codes," https://spherical-codes.org/.
[16] D. Sakrison, "A geometric treatment of the source encoding of a Gaussian random variable," *IEEE Trans. on Inf. Th.*, vol. 14, no. 3, 1968.
[17] T. Ericson and V. Zinoviev, *Codes on Euclidean spheres*. Elsevier, 2001.
[18] J. Conway and N. Sloane, *Sphere packings, lattices and groups, 3rd Ed.* Springer, 1998.
[19] J.-P. Adoul and M. Barth, "Nearest neighbor algorithm for spherical codes from the Leech lattice," *IEEE Trans. Inf. Theory*, vol. 34, no. 5, pp. 1188–1202, 1988.
[20] J. Schalkwijk, "An algorithm for source coding," *IEEE Trans. Inf. Theory*, vol. 18, no. 3, pp. 395–399, 1972.
[21] T. Muller, S. Ragot, V. Barriac, and P. Scalart, "Evaluation of objective quality models on neural audio codecs," in *Proc. IWAENC*, 2024.
[22] ITU-T Rec. P.863, "Perceptual objective listening quality prediction," Mar. 2018.
[23] Descript, "DAC," https://github.com/descriptinc/descript-audio-codec.
[24] J. Richter *et al.*, "EARS: An anechoic fullband speech dataset benchmarked for speech enhancement and dereverberation," in *ISCA Interspeech*, 2024, pp. 4873–4877.