QINCODEC: Neural Audio Compression with Implicit Neural Codebooks

Zineb Lahrichi, Gaëtan Hadjeres, Gael Richard, Geoffroy Peeters

QINCODEC: Neural Audio Compression with Implicit Neural Codebooks

Zineb Lahrichi
Sony AI, Télécom Paris
Paris, France

Gaëtan Hadjeres
Sony AI
Paris, France

Gaël Richard
Télécom Paris
Paris, France

Geoffroy Peeters
Télécom Paris
Paris, France

***Abstract*—Neural audio codecs, neural networks which compress a waveform into discrete tokens, play a crucial role in the recent development of audio generative models. State-of-the-art codecs rely on the end-to-end training of an autoencoder and a quantization bottleneck. However, this approach restricts the choice of the quantization methods as it requires to define how gradients propagate through the quantizer and how to update the quantization parameters online. In this work, we revisit the common practice of joint training and propose to quantize the latent representations of a pre-trained autoencoder offline, followed by an optional finetuning of the decoder to mitigate degradation from quantization. This strategy allows to consider any off-the-shelf quantizer, especially state-of-the-art trainable quantizers with implicit neural codebooks such as QINCO2. We demonstrate that with the latter, our proposed codec termed QINCODEC, is competitive with baseline codecs while being notably simpler to train. Finally, our approach provides a general framework that amortizes the cost of autoencoder pretraining, and enables more flexible codec design.**

***Index Terms*—Audio codecs, neural quantization**

I. INTRODUCTION

Traditional audio codecs, such as MP3 or Opus, are compression systems that encode digital audio into smaller bit-based intermediate representations, while enabling accurate reconstruction via a decoder. In recent years, neural audio codecs have emerged as robust alternatives to traditional handcrafted approaches and are now essential to the creation of speech and audio autoregressive generative models [1], [2].

Many advances in Neural Audio Codecs research are building upon the approach from [3] often denoted as RVQ-GAN, which consists in the end-to-end training of an autoencoder with a Residual Vector Quantization (RVQ) bottleneck layer. However, one critical assumption within this framework has remained unchallenged: namely the need for *end-to-end* training, which requires defining 1. how to propagate gradients (the quantization layer is non-differentiable by definition) and 2. how to perform online updates of the quantizer parameters, usually done via bespoke formulas.

These constraints significantly narrow the range of quantization layer designs since such operations may not always be possible. In particular, QINCO [4] and its follow-up QINCO2 [5] were proposed as powerful generalizations of unsupervised RVQ, relying on a trainable neural network that adapts codebooks implicitly for more accurate quantization. However, their complexity and specific training loop limit their suitability for online settings, as they are primarily designed for fixed dataset compression and retrieval. In this paper, we propose a three-stages strategy that allows us to rely on QINCO2 for neural audio coding:

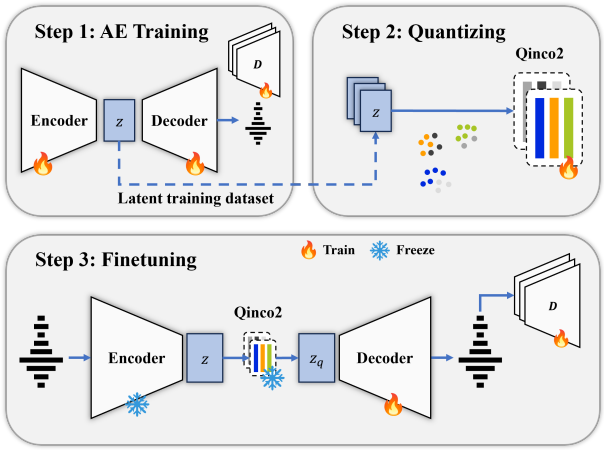

Fig. 1. **Training procedure of QINCODEC with offline quantization**: First, we train a continuous compression model with spectral and adversarial losses. Next, we quantize the bottleneck latent vectors into discrete embeddings. We then finetune the decoder on the quantized representations.

- We introduce QINCODEC, a 44.1 kHz audio codec based on the decoupled training of an autoencoder and a neural residual vector quantizer QINCO2, trained offline.
- Our model is the first auto-encoder that relies on Vocos [6] blocks, providing a lightweight and fast way to encode/decode audio, making its integration easy into the training pipelines of generative models.
- QINCODEC outperforms state-of-the-art methods at 16 kbps bitrate and achieves competitive results at 8kbps. with both objective and subjective metrics.
- Our offline approach offers a simple yet robust framework that allows to consider any off-the-shelf quantizer with a fixed pre-trained autoencoder, paving the way for adaptable and frugal codec design.

We believe that this work is the first to demonstrate the viability of non-end-to-end training for audio codecs and hope that this may drive attention to research in offline quantization methods and their applications for generative modeling. A website with audio examples is available at https://zinebl-sony.github.io/post-training-rvq/.

II. Related Work

A. Vector Quantization techniques

Vector Quantization (VQ) is the task of encoding continuous data into a discrete set of vector codes and is fundamental to data compression [7] and approximate nearest neighbor search [8]. Traditional VQ employs classical clustering algorithms, but its computational cost scales linearly with the number of codes, prompting the need for low-distortion alternatives. Multi-codebook approaches, such as Residual Vector Quantization (RVQ) [9], alleviate this by progressively refining quantization from coarse to fine. Lastly, neural quantizers have leveraged deep networks to compress data more efficiently—targeting large-scale database compression and rapid retrieval. Notably, Unsupervised Neural Quantization [10] combines deep networks with Product Quantization and the Gumbel-softmax trick, while QINCO [11] adapts residual codes within the RVQ framework.

B. Neural Audio Codecs

Recent advances in neural audio coding take inspiration from earlier approaches for learning discrete representations using deep neural autoencoders. A prominent class of these methods leverages continuous noisy relaxations of discrete variables such as Concrete relaxations [12] using the Gumbel-Softmax distribution [13], enabling optimization via backpropagation. Alternatively, Softmax quantization was also considered for speech coding [14].

These methods were progressively replaced by the widely adopted Vector Quantization Variational Autoencoders (VQ-VAEs). This class of models utilizes a straight-through estimator to pass gradients from the decoder to the encoder [7], together with Exponential Moving Average (EMA) updates of the codes based on their assigned vectors.

In the audio compression domain, [15] introduced RVQ-GANs, by integrating RVQ in the VAE-GAN framework [16]. This hierarchical quantization approach enables raw waveform compression at variable bitrates, while ensuring high-quality synthesis through the combined use of adversarial, reconstruction, and codebook losses. Multiple approaches [17], [18] improved upon this RVQ-GAN framework by addressing some of its major limitations such as low codebook usage, loss balancing and difficult hyperparameter tuning. This can be tackled, for instance, with architectural improvements [19], alternative quantization methods [20], propagating gradients with a variant of the straight-through estimator [21], and more sophisticated discriminators [22].

Now, the most recent development in audio compression involves scaling encoder-decoder architectures by incorporating transformer blocks before and after the quantization layer while retaining RVQ, as seen in models like the Mimi codec [2] or by using Finite Scalar Quantization (FSQ) [23].

III. Residual Vector Quantization (RVQ)

A. Conventional RVQ

Vector Quantization refers to the mapping of continuous data embeddings to discrete vectors selected from a finite set, called a *codebook*, with its size specified in bits. To this end, clustering heuristics like k-means [24] are used, but these methods exhibit poor scalability when applied to large codebooks. RVQ overcomes this limitation by using a sequence of smaller codebooks, where the residual quantization error is iteratively quantized by the next codebook. As in [4], we set some notations to formalize RVQ and its variants throughout the paper. We aim to quantize the vectors $x \in \mathbb{R}^D$ by using a sequence of codebooks $(\bar{C}_1, \dots, \bar{C}_N)$, each containing K entries $(\bar{c}_i^1, \dots, \bar{c}_i^K)_{i \in \{1,N\}}$. Let $\hat{x}_n$ be the reconstruction after $n-1$ steps, with $\hat{x}_1 = 0$. Quantization proceeds iteratively, where at each step n, the residual vector $r_n = x - \hat{x}_n$ is encoded by selecting the closest entry from $\bar{C}_n$.

B. QINCO: Implicit Neural Codebooks

A fundamental limitation of RVQ, is its use of static codebooks at each quantization stage, ignoring residual error distributions shaped by earlier codebooks. To address that, they introduce a neural residual vector quantizer that depends on prior intermediate reconstructions, and where the codebooks are learned *implicitely* via a neural network.

At each quantization step n, a neural network f_{θ_n} generates a specialized codebook centroid $c_n^k = f_{\theta_n}(\hat{x}_n, \bar{c}_n^k)$, conditioned on the previous reconstruction $\hat{x}_n$ and an initial centroid $\bar{c}_n^k$ taken from a pre-trained base codebook $\bar{C}_n$ obtained with k-means clustering as described in III-A.

For each centroid, an affine transformation projects the concatenation of $\bar{c}_n^k$ and $\hat{x}_n$, followed by L residual blocks. In addition, they set f_{θ_1} to identity to compensate the null conditioning $\hat{x}_1$ at the initial step, resulting in $C_1 = \bar{C}_1$. Residual connections enable base codebooks $(\bar{C}_1 ... \bar{C}_n)$ to propagate through the network, allowing it to achieve and outperform conventional RVQ baselines.

The conditioning procedure forces the decoding to be sequential with the update rule $\hat{x}_{n+1} \leftarrow \hat{x}_n + f_{\theta_n}(\hat{x}_n, \bar{c}_n^k)$. During training the parameters are learned via stochastic gradient descent to minimize the sum of mean squared errors between the residuals and centroids over all centroids and codebooks.

C. Improved Residual Vector Quantization (iRVQ)

We propose a straightforward variant of RVQ, by adding residual conditioning as in QINCO, relying on residual statistics and re-standardization. This quantization, which we introduce for our ablation study, can be seen as a special case of QINCO without neural network training.

More precisely, at each step n, the residual r_{n+1} after quantization of r_n is

$$r_{n+1} = (r_n - c)/\sigma_c \quad \text{with} \quad c := \underset{\bar{c}_n^k \in \bar{C}_n, k \in \{1 \dots K\}}{\arg\min} ||r_n - \bar{c}_n^k||_2^2 \tag{1}$$

where $\sigma_c \in \mathbb{R}^D$ is the per-cluster standard deviation of all inputs assigned to cluster c. The final approximation $\hat{x}$ can be reconstructed with the update rule $\hat{x}_{n+1} \leftarrow \hat{x}_n + \sigma_c c$, without re-centering to ensure decreasing quantization errors with respect to the number of codebooks. It should be noted that

such standardization procedure may be complex to implement in end-to-end autoencoder training as one would have to update these statistics online.

Null codebook Since RVQ is additive, we fix a null vector to each codebook (except the first one) to ensure the decrease of the quantization error after each quantization step in RVQ and avoid the addition of random noise.

IV. QINCODEC

A. Step 1: Autoencoder pre-training

Our proposed autoencoder borrows ideas from Vocos [6], a GAN-based vocoder, trained to produce STFT coefficients from an audio signal. The decoder is a succession of ConvNeXt blocks followed by an iSTFT head and our encoder consists in its mirrored architecture namely a complex STFT layer followed by ConvNeXt blocks. Finally, a linear bottleneck is inserted between the encoder and the decoder. The primary advantage of this architecture is its consistent temporal resolution across all layers, avoiding artifacts from upsampling [25]. Additionally, since convolutions are performed on sequences of uniform length $\text{sample size}/\text{hop length}$, the architecture ensures fast training.

The discriminators we use are identical to the ones used in [17] and our objective includes a spectral loss, an adversarial loss and a feature matching loss. More details are provided below in V-A

B. Step 2: Offline Vector Quantization

After training the compression model, we leverage its latent representations to train a residual vector quantizer in an offline setting. We consider a set of audio embeddings $\mathcal{Z}$ deriving from our encoder $\mathcal{E}$ and audio inputs. Each embedding lies in $\mathbb{R}^{D\times T}$, where T and D are respectively the number of frames and the latent dimension of our model. From $\mathcal{Z}$, we form a collection $\mathcal{X}$ of vector frames in $\mathbb{R}^D$, which are subsequently used to train the residual vector quantizer. In this case, the number of quantizers N determines the target $\text{bitrate} = N \times \log_2 K \times F$, where $F = T/d$ is the latent frame rate, d the duration of the input audio and $\log_2 K$ is the size of each codebook in binary bits. The offline vector quantization step is generally lightweight and only takes a fraction of the time of the pre-training or finetuning stages.

C. Step 3: Finetuning

In the finetuning stage, we freeze the encoder and the quantizer and back-propagate only through the decoder and the discriminators. In other words, we improve the previously trained GAN-vocoder in accomplishing the task of decoding the quantized representation.

V. EXPERIMENTAL SETUP

A. Training details

For the first step of our framework, we train three compression models with latent dimension $D = \{16, 32, 64\}$, with the architecture described in IV-A. Our three models are trained using the AdamW optimizer with weight decay of 1e^{-3} and a learning rate of 2e^{-4} including exponential warmup and decay. Betas are set to $(0.5, 0.9)$. Each model is trained for 1M steps across 8 A100 GPUs, with an effective batch size of 240.

Our inputs are complex STFTs, configured with $\text{n}_{\text{fft}} = 512$, a hop length of 256 and input audio duration is $d = 1$s. As in the original Vocos model, we use 8 ConvNeXt layers for both the encoder and the decoder. For the discriminator, we combine the complex multi-scale STFT and the Multi-Period Discriminator (MPD), as in [17]. The training objective includes a spectral reconstruction L_1 loss based on mel-coefficients as in [17], an adversarial loss using the LSGAN formulation [26], and a feature matching loss as formulated in [17]. The corresponding weights for these losses are set to 15, 1, and 2, respectively.

For the finetuning step, we keep the same configuration as for training, except for the learning rate which is aligned with the learning rate of the last training epoch.

B. Baselines

We evaluate the performance of QINCODEC against two baselines: the pretrained 44.1kHz DAC and 48kHz ENCODEC models at 8kbps and 16kbps. Both are RVQ-GANs, following the same framework as [3] and trained on general sounds among other types like speech and music. However, they differ in their codebook update strategies, loss-balancing techniques, and architectural designs.

C. Datasets

We train and validate our models on 1-second, 44.1KHz audio clips from WavCaps [27], a dataset of 400k general sounds from AudioSet, FreeSound, BBC Sound Effects, and SoundBible. For quantization, we randomly sample audios from the training set, extract their latent representations, and shuffle them. We use around 5M frames for conventional RVQ and 60M frames for training QINCO2. Evaluation is conducted with 5-second audio chunks from the AudioCaps [28] test set, ensuring no overlap with the training data and consistency with the baselines evaluation protocols.

D. Objective and subjective metrics

We evaluate reconstruction quality using objective and perceptual metrics. For objective metrics, we measure the Scale-Invariant Signal-to-Distortion Ratio (Si-SDR) and multi-scale mel reconstruction error, as defined in V-A. For perceptual metrics, we compute the Fréchet Distance and feature matching with OpenL3 [29]. To assess quantization quality in the bottleneck, we measure the Perplexity i.e the entropy of the codebook's representation, which reflects how uniformly codebook entries are used.

We conducted a MUSHRA test [1] to compare the perceptual quality of QINCODEC with DAC at 8 and 16 kbps. The test included 12 randomly selected 5-second excerpts spanning various sound categories, with a hidden reference and a 3.5 kHz low-pass filtered anchor.

[1] https://github.com/audiolabs/webMUSHRA

TABLE I
QUANTITATIVE EVALUATION OF THE PROPOSED MODEL AT 16 KBPS COMPARED TO COMPETING BASELINES.

	Quantizer	Finetuning	Si-SDR ↑	MS-Mel ↓	FD-OL3 ↓	FM-OL3 ↓	Perplexity ↑
DAC			**9.04**	0.85	51.4	0.23	784
EnCodec			6.10	1.39	97.2	0.35	588
	iRVQ	✗	6.20	0.93	45.5	0.23	926
	iRVQ	✓	6.58	0.82	31.3	0.21	926
	QINCO2	✗	7.22	0.79	38.1	0.21	980
QINCODEC	QINCO2	✓	7.55	**0.74**	**34.4**	**0.19**	**980**

TABLE II
QUANTITATIVE EVALUATION OF THE PROPOSED MODEL AT 8 KBPS COMPARED TO COMPETING BASELINES.

	Quantizer	Finetuning	Si-SDR ↑	MS-Mel ↓	FD-OL3 ↓	FM-OL3 ↓	Perplexity ↑
DAC			**5.49**	**1.07**	45.7	0.28	746
EnCodec			3.22	1.57	98.4	0.39	506
	iRVQ	✗	2.60	1.46	77.0	0.33	919
	iRVQ	✓	3.77	1.14	39.3	0.28	919
	QINCO2	✗	3.96	1.32	67.1	0.31	957
QINCODEC	QINCO2	✓	4.64	1.12	**35.7**	**0.27**	**957**

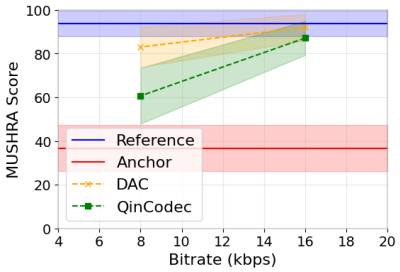

Fig. 2. MUSHRA scores with 95% confidence intervals for DACand QINCODEC and a 3.5kHz low-pass anchor, evaluated at 8 kbps and 16 kbps.

E. Quantization details

RVQ and iRVQ: We utilize the mini batched implementation of k-means[2] to scale to a few million vectors. We adapted the batch size to three times the size of the codebooks and otherwise keep the default parameters.

QINCO2: The original paper provides detailed guidance on how to select hyperparameters that optimize various quantizer attributes, including precision (low MSE), encoding/decoding efficiency, and training speed. For our experiments, we prioritize a balanced tradeoff between training efficiency and reconstruction accuracy, selecting a medium-sized model with $L = 4$ residual blocks and a hidden dimension of $d_e = d_h = 384$. Regarding optimization, we use the default parameters specified in the original paper.

VI. RESULTS AND DISCUSSIONS

A. Comparison with baselines

QINCODEC outperforms DAC and ENCODEC at 16 kbps across all metrics except Si-SDR, where DAC scores higher (see Table I and Fig.2), likely due to the phase information loss in the quantization layer. Fig.3 shows that QINCODEC achieves higher and more consistent perplexity scores across codebooks, highlighting the benefit of offline-trained quantizers for codebook optimization. At 8 kbps, the gap between DAC and QINCODEC narrows, leading to similar performance (see Table II).

MUSHRA scores align with SI-SDR, with DAC slightly ahead but within overlapping confidence intervals. Both models surpass the 3.5 kHz low-pass anchor. While QINCODEC matches or exceeds baselines in objective metrics, its perceived quality remains limited by the quantizer capacity, particularly at lower bitrates.

B. Ablation studies

Table I shows the perceptual improvements gained from finetuning after offline quantization. A few extra epochs of finetuning enhance perceptual metrics like FD-OL3, with minimal impact on spectral metrics, highlighting the importance of finetuning, even briefly, to optimize perceptual quality.

We also evaluate our model with different offline quantizers: iRVQ and QINCO2 (see Table III). iRVQ improves code precision over RVQ, likely by handling outliers through re-standardization. QINCO2 outperforms both, with gains reflected in audio metrics at 8 and 16 kbps, demonstrating that a more expressive codebook enhances reconstruction quality.

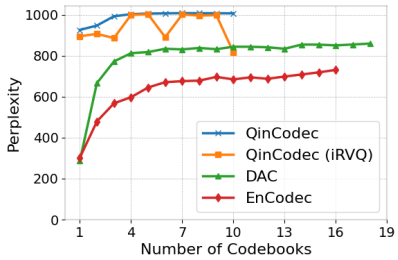

Fig. 3. Perplexity vs. number of codebooks for ENCODEC, DAC, QINCODEC, and QINCODEC with iRVQ quantization.

[2]https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html

TABLE III
PERFORMANCE OF OFFLINE QUANTIZERS AT 16 KBPS, AFTER STEP 2.

	MSE (z) ↓	Si-SDR ↑	MS-Mel ↓
RVQ	2.29	6.09	0.96
iRVQ	2.23	6.20	0.93
QINCO2	**1.51**	**7.22**	**0.79**

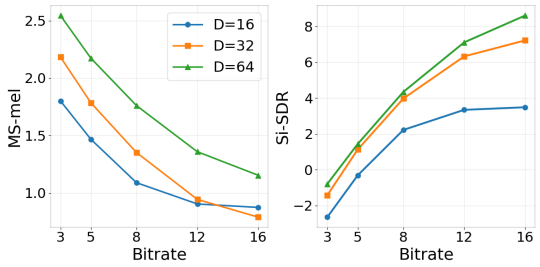

Fig. 4. Performance of QINCODEC at various bitrates, and latent dimensions.

C. Influence of the latent dimension

Increasing the latent dimension D improves reconstruction fidelity by embedding more information in continuous models, but these gains do not directly translate to offline quantization, which is inherently limited by the bitrate. Fig. 4 shows that higher D increases MS-mel (since quantization becomes more complex and lossy), but also improves SI-SDR by encoding richer embeddings, including phase information. In conclusion, the latent dimension D represents the trade-off between fidelity and compression rate and has to be chosen carefully (In our case, $D = 32$ seems to be the optimal choice).

VII. CONCLUSION

We show that end-to-end training is not a prerequisite for vector-quantized neural audio compression. Our codec, QINCODEC, leverages a novel three-step training procedure with offline quantization to eliminate complex gradient propagation while enhancing quantization performance. This approach allows to use any off-the-shelf quantizer like QINCO without optimization constraints, yielding competitive results against end-to-end baselines. However, a trade-off between distortion, compression rate, and latent dimension limits performance at lower bitrates. In future work, we will narrow the gap between end-to-end and modular approaches and develop offline quantizers optimized for audio coding, capitalizing on the training stability of our method to scale Neural Audio Coding architectures.

REFERENCES

[1] Z. Borsos, R. Marinier, D. Vincent, E. Kharitonov, O. Pietquin, M. Sharifi, D. Roblek, O. Teboul, D. Grangier, M. Tagliasacchi, and N. Zeghidour, "Audiolm: a language modeling approach to audio generation," 2023. https://arxiv.org/abs/2209.03143

[2] A. Défossez, L. Mazaré, M. Orsini, A. Royer, P. Pérez, H. Jégou, E. Grave, and N. Zeghidour, "Moshi: a speech-text foundation model for real-time dialogue," *arXiv preprint arXiv:2410.00037*, 2024.

[3] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "Soundstream: An end-to-end neural audio codec," 2021. https://arxiv.org/abs/2107.03312

[4] I. A. Huijben, M. Douze, M. Muckley, R. J. Van Sloun, and J. Verbeek, "Residual quantization with implicit neural codebooks," *arXiv preprint arXiv:2401.14732*, 2024.

[5] T. Vallaeys, M. Muckley, J. Verbeek, and M. Douze, "Qinco2: Vector compression and search with improved implicit neural codebooks," 2025. https://arxiv.org/abs/2501.03078

[6] H. Siuzdak, "Vocos: Closing the gap between time-domain and fourier-based neural vocoders for high-quality audio synthesis," 2024. https://arxiv.org/abs/2306.00814

[7] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *CoRR*, 2017. http://arxiv.org/abs/1711.00937

[8] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization for approximate nearest neighbor search," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2946–2953.

[9] J. Martinez, H. H. Hoos, and J. J. Little, "Stacked quantizers for compositional vector compression," *CoRR*, 2014. http://arxiv.org/abs/1411.2173

[10] S. Morozov and A. Babenko, "Unsupervised neural quantization for compressed-domain similarity search," 2019. https://arxiv.org/abs/1908.03883

[11] I. A. M. Huijben, M. Douze, M. Muckley, R. J. G. van Sloun, and J. Verbeek, "Residual quantization with implicit neural codebooks," 2024. https://arxiv.org/abs/2401.14732

[12] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," 2017. https://arxiv.org/abs/1611.00712

[13] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2017. https://arxiv.org/abs/1611.01144

[14] S. Kankanahalli, "End-to-end optimized speech coding with deep neural networks," *CoRR*, 2017. http://arxiv.org/abs/1710.09064

[15] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "Soundstream: An end-to-end neural audio codec," *CoRR*, 2021. https://arxiv.org/abs/2107.03312

[16] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," 2016. https://arxiv.org/abs/1512.09300

[17] R. Kumar, P. Seetharaman, A. Luebs, I. Kumar, and K. Kumar, "High-fidelity audio compression with improved rvqgan," 2023. https://arxiv.org/abs/2306.06546

[18] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," 2022. https://arxiv.org/abs/2210.13438

[19] S. Ahn, B. J. Woo, M. H. Han, C. Moon, and N. S. Kim, "Hilcodec: High fidelity and lightweight neural audio codec," 2024. https://arxiv.org/abs/2405.04752

[20] F. Mentzer, D. Minnen, E. Agustsson, and M. Tschannen, "Finite scalar quantization: Vq-vae made simple," 2023. https://arxiv.org/abs/2309.15505

[21] C. Fifty, R. G. Junkins, D. Duan, A. Iger, J. W. Liu, E. Amid, S. Thrun, and C. Ré, "Restructuring vector quantization with the rotation trick," *arXiv preprint arXiv:2410.06424*, 2024.

[22] T. Bak, J. Lee, H. Bae, J. Yang, J.-S. Bae, and Y.-S. Joo, "Avocodo: Generative adversarial network for artifact-free vocoder," 2023. https://arxiv.org/abs/2206.13404

[23] J. D. Parker, A. Smirnov, J. Pons, C. Carr, Z. Zukowski, Z. Evans, and X. Liu, "Scaling transformers for low-bitrate high-quality speech coding," *arXiv preprint arXiv:2411.19842*, 2024.

[24] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006.

[25] J. Pons, S. Pascual, G. Cengarle, and J. Serrà, "Upsampling artifacts in neural audio synthesis," *CoRR*, 2020. https://arxiv.org/abs/2010.14356

[26] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, and Z. Wang, "Multi-class generative adversarial networks with the L2 loss function," *CoRR*, 2016. http://arxiv.org/abs/1611.04076

[27] X. Mei, C. Meng, H. Liu, Q. Kong, T. Ko, C. Zhao, M. D. Plumbley, Y. Zou, and W. Wang, "Wavcaps: A chatgpt-assisted weakly-labelled audio captioning dataset for audio-language multimodal research," *IEEE/ACM TASLP*, vol. 32, p. 3339–3354, 2024. http://dx.doi.org/10.1109/TASLP.2024.3419446

[28] C. D. Kim, P. Kim, H. Lee, and G. Kim, "AudioCaps: Generating Captions for Audios in The Wild," in *NAACL-HLT*, 2019.

[29] A. L. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen, and learn more: Design choices for deep audio embeddings," in *IEEE ICASSP*, 2019, pp. 3852–3856.