QINCODEC : Quantificateurs Neuronaux pour la Compression Audio

Zineb LAHRICHI[1, 2] Gaëtan HADJERES[1] Gaël RICHARD[2] Geoffroy PEETERS[2]

[1]Télécom Paris

[2]Sony AI

Résumé – Les codecs audio à l'état de l'art reposent sur l'entrainement *de bout en bout* d'un autoencodeur et d'une quantification latente, contraignant le choix des méthodes de quantification (non différentiables par définition) et imposant leur apprentissage *en ligne*. Nous proposons une nouvelle méthode : quantifier *hors ligne* les représentations latentes d'un autoencodeur pré-entrainé, suivie d'un ré-entrainement du décodeur pour atténuer la dégradation. Cette stratégie permet d'utiliser n'importe quel quantificateur, y compris des quantificateurs neuronaux comme QINCO2, rendant la conception du codec plus flexible et moins coûteuse.

Abstract – State-of-the-art codecs rely on end-to-end training of an autoencoder with a quantization bottleneck. This approach limits quantization method choices due to the need to propagate gradients and update parameters online. In this work, we propose quantizing the latent representations of a pre-trained autoencoder offline, followed by optional decoder finetuning to reduce degradation. This strategy allows using any quantizer, including neural ones like QINCO2, making codec training simpler. Our approach provides a flexible framework that amortizes pretraining costs and enables more versatile codec design.

1 Introduction

Les codecs audio neuronaux, modèles de compression discrets, ont émergé comme des alternatives aux approches traditionnelles tels que MP3 ou Opus. Ils sont désormais essentiels à la création de modèles génératifs autoregressifs d'audio [3]. Les recherches récentes consistent à entrainer *de bout en bout* un autoencodeur avec une couche de Quantification Vectorielle Résiduelle (RVQ) [21]. Cependant, une hypothèse clé dans ce cadre n'a jamais été remise en question : la nécessité d'un entrainement *de bout en bout*, qui impose de définir comment propager les gradients (la couche de quantification est par définition non différentiable) et comment mettre à jour les paramètres du quantificateur, ce qui nécessite généralement une procédure d'entrainement plus complexe.

De telles opérations ne sont pas toujours réalisables. En particulier, QINCO [8] et son évolution QINCO2 [19] généralisent la RVQ de manière non supervisée, en s'appuyant sur un réseau neuronal capable d'adapter implicitement les codebooks pour une quantification plus précise. Cependant leur complexité limitent leur applicabilité en contexte *en ligne* car ils sont conçus principalement pour la compression et la recherche d'information sur des ensembles de données fixes.

Dans cet article, nous proposons QINCODEC, un codec audio à 44,1 kHz basé sur l'entrainement découplé d'un autoencodeur et d'un RVQ neuronal QINCO2, entrainé *hors ligne*. QINCODEC surpasse l'état de l'art à un débit de 16 kbps (*kilobit par seconde*) et obtient des résultats compétitifs à 8 kbps, en termes de métriques objectives et subjectives. Notre modèle est le premier autoencodeur à utiliser l'architecture Vocos [18], offrant une solution légère pour l'encodage/décodage audio. Enfin, notre approche *hors ligne* permet l'emploi de n'importe quel quantificateur avec un autoencodeur pré-entrainé, favorisant une conception de codec adaptable et moins coûteuse en ressources. A notre connaissance, ce travail est le premier à démontrer la viabilité de l'entrainement découplé des codecs audio. Une démonstration est disponible sur `https://zinebl-sony.github.io/post-training-rvq/`.

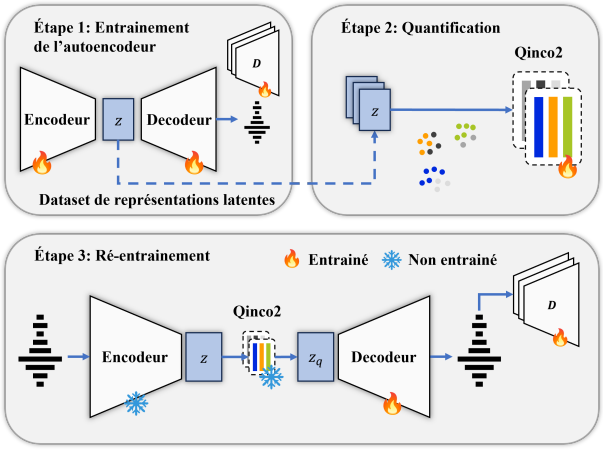

FIGURE 1 : **Procédure d'entrainement de QINCODEC** : nous entrainons d'abord un autoencodeur continue, puis nous quantifions sa représentation latente. Enfin, nous ré-entrainons le décodeur sur les représentations quantifiées.

2 Travaux Connexes

2.1 Techniques de quantification vectorielle

La Quantification Vectorielle (VQ) encode des données continues en un ensemble discret de codes et est essentielle pour la compression et la recherche des plus proches voisins [20]. Les méthodes classiques de VQ s'appuient sur des algorithmes de clustering, mais leur coût croit linéairement avec la taille des codes. La Quantification Vectorielle Résiduelle (RVQ) [14] contourne cette limitation en raffinant progressivement la quantification. Les quantificateurs neuronaux, comme la quantification neuronale non supervisée [16] et QINCO [7], exploitent des réseaux neuronaux profonds pour une compression plus efficace.

2.2 Codecs audio neuronaux

Les codecs audio neuronaux récents exploitent l'apprentissage profond en associant des autoencodeurs et la quantification vectorielle. La relaxation Concrete [12] et la distribution Gumbel-Softmax [9] permettent l'optimisation par rétropropagation du gradient à travers le quantificateur. Les auto-encodeurs variationnel quantifiés vectoriellement (ou VQ-VAE) utilisent un estimateur direct pour la propagation des gradients et des mises à jour exponentielles des codes [20]. Les RVQ-GANs [22] intègrent la RVQ dans une architecture adversariale (VAE-GAN), offrant une compression à débit variable avec une synthèse de haute qualité grâce à un objectif adversarial et spectral. Les progrès récents incluent l'amélioration des dictionnaires de quantification (ou *codebooks*) [11, 5], l'équilibrage des fonctions de cout et des hyperparamètres avec des modifications architecturales [1], des variantes d'estimateurs directs [6] et des discriminateurs avancés [2].

3 Quantification Vectorielle Résiduelle

3.1 RVQ conventionnelle

La VQ consiste à associer des représentations continues à des vecteurs discrets issus d'un codebook fini, dont la taille est spécifiée en bits. Cependant, les heuristiques de clustering utilisées pour la quantification deviennent inadaptées pour de grands codebooks. La RVQ surmonte cette limite en utilisant une séquence de codebooks plus petits, où l'erreur est itérativement quantifiée par le codebook suivant. Comme formulé dans [8], nous cherchons à quantifier les vecteurs $x \in \mathbb{R}^D$ à l'aide d'une séquence de codebooks $(\bar{C}_1, \ldots, \bar{C}_N)$, chacun contenant K entrées ou centroïdes $(\bar{c}_i^1, \ldots, \bar{c}_i^K)_{i\in\{1,N\}}$. Soit $\hat{x}_n$ la reconstruction après $n-1$ étapes, avec $\hat{x}_1 = 0$. À chaque étape n, le vecteur résiduel $r_n = x - \hat{x}_n$ est encodé en sélectionnant l'entrée la plus proche de $\bar{C}_n$.

3.2 QINCO : Codebooks neuronaux implicites

Une limitation de RVQ est l'utilisation de codebooks statiques à chaque étape, ignorant la distribution des erreurs résiduelles. QINCO introduit un quantificateur résiduel neuronal qui génère des centroïdes en fonction des reconstructions précédentes. Un réseau f_{θ_n} ajuste chaque centroïde comme : $c_n^k = f_{\theta_n}(\hat{x}_n, \bar{c}_n^k)$, où $\bar{c}_n^k$ provient d'un codebook pré-entrainé par k-means. Chaque centroïde est obtenu par une transformation affine suivie de L blocs résiduels. Pour la première étape, f_{θ_1} est fixé à l'identité, soit $C_1 = \bar{C}_1$. La mise à jour suit : $\hat{x}_{n+1} \leftarrow \hat{x}_n + f_{\theta_n}(\hat{x}_n, \bar{c}_n^k)$. Enfin, l'entrainement optimise par descente de gradient stochastique la somme des erreurs quadratiques moyennes entre résidus et centroïdes.

3.3 Quantification Vectorielle Résiduelle Améliorée (iRVQ)

Nous proposons une variante simplifiée de RVQ inspirée de QINCO, sans apprentissage neuronal, basée sur la restandardisation des résiduels. Après quantification, les residuels sont mis à jour :

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

où $\sigma_c \in \mathbb{R}^D$ est l'écart-type des échantillons assignés à c. La reconstruction est ensuite calculée via $\hat{x}_{n+1} \leftarrow \hat{x}_n + \sigma_c c$. Comme cette procédure nécessite une mise à jour *en ligne* des statistiques, il est difficile de l'intégrer dans un entrainement *de bout en bout* de l'autoencodeur. RVQ étant additif, un vecteur nul est ajouté à chaque codebook (sauf le premier) pour réduire l'erreur de quantification et éviter le bruit aléatoire.

4 QINCODEC

L'entrainement de QINCODEC est réalisé en 3 étapes :

1. **Entrainement de l'autoencodeur.** L'autoencodeur s'inspire de Vocos [18], un vocodeur GAN qui génère un spectrogramme à partir d'un signal audio. Le décodeur combine des blocs ConvNeXt et une Transformée de Fourier Court Terme (TFCT) inverse tandis que l'encodeur suit une architecture symétrique avec une TFCT complexe et des blocs ConvNeXt. L'architecture maintient une résolution temporelle constante, limitant les artefacts de suréchantillonnage [17]. Les convolutions sont appliquées sur des fenêtres de longueur fixe accélérant l'entrainement. Enfin, les discriminateurs suivent [11], avec un objectif spectral, adversarial et de correspondance (*feature matching*).

2. **Quantification *hors ligne*.** Après l'entrainement de l'autoencodeur, nous exploitons ses représentations latentes pour entrainer un quantificateur vectoriel résiduel *hors ligne*. Les embeddings audio $\mathcal{Z} \in \mathbb{R}^{D\times T}$ issus de l'encodeur sont définis pour T trames audios et une dimension latente D. La collection de trames vectorielles dans $\mathbb{R}^D$ est utilisée pour entrainer le quantificateur. Le nombre de quantificateurs N fixe le débit cible : $\textit{Débit} = N \times \log_2 K \times F, \quad F = T/d$ où d est la durée de l'audio et $\log_2 K$ est la taille du codebook en bits.

3. **Ré-entrainement du décodeur.** Lors du ré-entrainement, les poids de l'encodeur et du quantificateur ne sont pas modifiés. Seuls le décodeur et les discriminateurs sont optimisés, améliorant ainsi la capacité du vocodeur GAN à décoder la représentation quantifiée.

5 Protocole expérimental

Nous entrainons trois modèles de compression (comme décrits en 4) avec une dimension latente $D = \{16, 32, 64\}$ et un optimiseur AdamW ($\text{lr} = 2 \cdot 10^{-4}, \text{batchsize} = 30$) pendant 1M d'étapes sur 8 GPUs A100. L'objectif inclut un objectif spectral L_1 sur les coefficients mel, un objectif adversarial avec LSGAN [13], et un objectif *feature matching*, avec des poids de 15, 1 et 2. le ré-entrainement suit la même configuration, avec un taux d'apprentissage fixé au dernier epoch d'entrainement. Pour la quantification, nous utilisons l'algorithme k-means (mini-batch) pour RVQ et iRVQ, et QINCO2, est entrainé avec $L = 4$, $d_e = d_h = 384$.

TABLE 1 : Évaluation quantitative du modèle proposé à 16 kbps comparé à des baselines.

	quantificateur	ré-entrainement	Si-SDR ↑	MS-Mel ↓	FD-OL3 ↓	FM-OL3 ↓	Perplexité ↑
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 2 : Évaluation quantitative du modèle proposé à 8 kbps comparé à des baselines.

	quantificateur	ré-entrainement	Si-SDR ↑	MS-Mel ↓	FD-OL3 ↓	FM-OL3 ↓	Perplexité ↑
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**

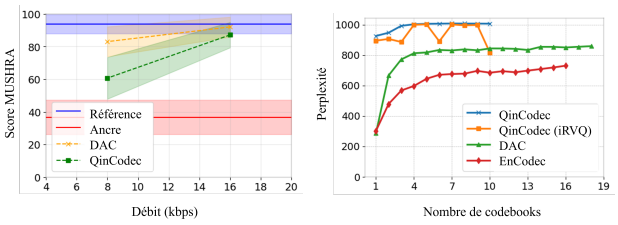

FIGURE 2 : (Gauche) Scores MUSHRA avec intervalles de confiance à 95% pour DAC et QINCODEC (Droite) Perplexité vs. nombre de codebooks pour chaque baseline.

5.1 Baselines et données

Nous comparons QINCODEC à deux baselines : les modèles pré-entrainés DAC (44.1kHz) et ENCODEC (48kHz) à 8kbps et 16kbps. Les deux sont des RVQ-GANs basés sur [21], avec des différences dans la mise à jour des codebooks, la pondération des objectifs et l'architecture. Nous entrainons nos modèles sur des extraits audio de 1s à 44.1kHz du dataset WavCaps [15], comprenant 400k sons généraux (AudioSet, FreeSound, BBC Sound Effects et SoundBible). Pour la quantification, nous échantillonnons aléatoirement des représentations latentes de l'ensemble d'entrainement. Nous utilisons environ 5M de trames pour RVQ et 60M pour QINCO2. L'évaluation est réalisée sur des extraits de 5s du jeu de test AudioCaps [10], sans chevauchement avec l'ensemble d'entrainement.

5.2 Metriques objectives et subjectives

La qualité de reconstruction est évaluée avec des métriques objectives (Si-SDR, erreur de reconstruction mel multi-échelle, Distance de Fréchet et feature matching avec OpenL3) [4]). Pour évaluer la qualité de la quantification, nous mesurons la perplexité, c'est-à-dire l'entropie de la représentation des codebooks. Un test MUSHRA a été réalisé pour comparer la qualité perceptuelle de QINCODEC avec DAC à 8 et 16 kbps. Le test inclut 12 extraits aléatoires de 5s couvrant diverses catégories sonores, avec une référence et une version filtrée passe-bas à 3.5 kHz (ancre).

6 Résultats et discussions

1. **Comparaison avec les modèles de référence**. QINCODEC surpasse DAC et ENCODEC à 16 kbps sur toutes les métriques, sauf le Si-SDR, où DAC obtient un score supérieur (voir tableau 1 et Fig. 2), probablement dû à une perte de phase lors de la quantification. La Fig. 2 (droite) montre que QINCODEC atteint une perplexité plus élevée et plus stable, soulignant l'apport des quantificateurs *hors ligne* à l'optimisation du codebook. À 8 kbps, l'écart entre DAC et QINCODEC se réduit, avec des performances comparables (voir tableau 2). Les scores MUSHRA sont corrélés au Si-SDR, avec DAC légèrement meilleur mais dans l'intervalle de confiance. Les deux modèles dépassent l'ancre passe-bas à 3,5 kHz.

2. **Études d'ablation** Le tableau 1 montre que le ré-entrainement après quantification *hors ligne* améliore les métriques perceptuelles, avec un impact minimal sur les métriques spectrales. Nous évaluons également différents quantificateurs *hors ligne* avant l'étape de ré-entrainement : iRVQ et QINCO2 (voir tableau 3). iRVQ améliore la précision de l'encodage, probablement grâce à la standardisation des valeurs aberrantes. QINCO2 surpasse les deux, avec des gains sur les métriques audio à 8 et 16 kbps, confirmant qu'un codebook plus expressif améliore la reconstruction.

3. **Influence de la dimension latente** L'augmentation de la dimension latente D améliore la fidélité de reconstruction en intégrant plus d'informations dans les modèles continus, mais ces gains sont limités par le débit lors de la quantification *hors ligne*. La Fig. 3 montre que des valeurs élevées de D augmentent MS-mel (révélant une quantification plus complexe et plus perdue) tout en améliorant le SI-SDR en capturant des embeddings plus riches, y compris l'information de phase. Ainsi, D représente un compromis entre fidélité et taux de compression ; $D = 32$ semble optimal dans notre cas.

TABLE 3 : Performance des quantificateurs à 16 kbps.

	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**

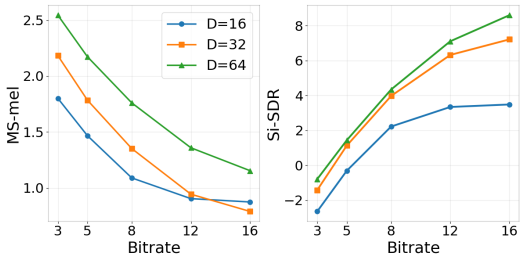

FIGURE 3 : Performance de QINCODEC à différents débits et dimensions latentes.

7 CONCLUSION

Dans cet article, nous tentons de démontrer que l'entrainement *de bout en bout* n'est pas nécessaire pour les codecs audio. QINCODEC adopte une procédure en trois étapes avec une quantification *hors ligne*, simplifiant l'optimisation tout en offrant des performances compétitives. Toutefois, un compromis entre distorsion, taux de compression et dimension latente limite la qualité à faibles débits. Les travaux futurs exploreront l'amélioration des quantificateurs *hors ligne* et la réduction de l'écart avec les approches *de bout en bout*, en tirant parti de la stabilité de notre méthode.

Références

[1] Sunghwan AHN, Beom Jun WOO, Min Hyun HAN, Chanyeong MOON et Nam Soo KIM : Hilcodec : High fidelity and lightweight neural audio codec, 2024.

[2] Taejun BAK, Junmo LEE, Hanbin BAE, Jinhyeok YANG, Jae-Sung BAE et Young-Sun JOO : Avocodo : Generative adversarial network for artifact-free vocoder, 2023.

[3] Zalán BORSOS, Raphaël MARINIER, Damien VINCENT, Eugene KHARITONOV, Olivier PIETQUIN, Matt SHARIFI, Dominik ROBLEK, Olivier TEBOUL, David GRANGIER, Marco TAGLIASACCHI et Neil ZEGHIDOUR : Audiolm : a language modeling approach to audio generation, 2023.

[4] Aurora Linh CRAMER, Ho-Hsiang WU, Justin SALAMON et Juan Pablo BELLO : Look, listen, and learn more : Design choices for deep audio embeddings. *In IEEE ICASSP*, pages 3852–3856, 2019.

[5] Alexandre DÉFOSSEZ, Jade COPET, Gabriel SYNNAEVE et Yossi ADI : High fidelity neural audio compression, 2022.

[6] Christopher FIFTY, Ronald G JUNKINS, Dennis DUAN, Aniketh IGER, Jerry W LIU, Ehsan AMID, Sebastian THRUN et Christopher RÉ : Restructuring vector quantization with the rotation trick. *arXiv preprint arXiv :2410.06424*, 2024.

[7] Iris A. M. HUIJBEN, Matthijs DOUZE, Matthew MUCKLEY, Ruud J. G. van SLOUN et Jakob VERBEEK : Residual quantization with implicit neural codebooks, 2024.

[8] Iris AM HUIJBEN, Matthijs DOUZE, Matthew MUCKLEY, Ruud JG VAN SLOUN et Jakob VERBEEK : Residual quantization with implicit neural codebooks. *arXiv preprint arXiv :2401.14732*, 2024.

[9] Eric JANG, Shixiang GU et Ben POOLE : Categorical reparameterization with gumbel-softmax, 2017.

[10] Chris Dongjoo KIM, Pyeongchang KIM, Hyunmin LEE et Gunhee KIM : AudioCaps : Generating Captions for Audios in The Wild. *In NAACL-HLT*, 2019.

[11] Rithesh KUMAR, Prem SEETHARAMAN, Alejandro LUEBS, Ishaan KUMAR et Kundan KUMAR : High-fidelity audio compression with improved rvqgan, 2023.

[12] Chris J. MADDISON, Andriy MNIH et Yee Whye TEH : The concrete distribution : A continuous relaxation of discrete random variables, 2017.

[13] Xudong MAO, Qing LI, Haoran XIE, Raymond Y. K. LAU et Zhen WANG : Multi-class generative adversarial networks with the L2 loss function. *CoRR*, 2016.

[14] Julieta MARTINEZ, Holger H. HOOS et James J. LITTLE : Stacked quantizers for compositional vector compression. *CoRR*, 2014.

[15] Xinhao MEI, Chutong MENG, Haohe LIU, Qiuqiang KONG, Tom KO, Chengqi ZHAO, Mark D. PLUMBLEY, Yuexian ZOU et Wenwu WANG : Wavcaps : A chatgpt-assisted weakly-labelled audio captioning dataset for audio-language multimodal research. *IEEE/ACM TASLP*, 32:3339–3354, 2024.

[16] Stanislav MOROZOV et Artem BABENKO : Unsupervised neural quantization for compressed-domain similarity search, 2019.

[17] Jordi PONS, Santiago PASCUAL, Giulio CENGARLE et Joan SERRÀ : Upsampling artifacts in neural audio synthesis. *CoRR*, 2020.

[18] Hubert SIUZDAK : Vocos : Closing the gap between time-domain and fourier-based neural vocoders for high-quality audio synthesis, 2024.

[19] Théophane VALLAEYS, Matthew MUCKLEY, Jakob VERBEEK et Matthijs DOUZE : Qinco2 : Vector compression and search with improved implicit neural codebooks, 2025.

[20] Aäron van den OORD, Oriol VINYALS et Koray KAVUKCUOGLU : Neural discrete representation learning. *CoRR*, 2017.

[21] Neil ZEGHIDOUR, Alejandro LUEBS, Ahmed OMRAN, Jan SKOGLUND et Marco TAGLIASACCHI : Soundstream : An end-to-end neural audio codec, 2021.

[22] Neil ZEGHIDOUR, Alejandro LUEBS, Ahmed OMRAN, Jan SKOGLUND et Marco TAGLIASACCHI : Soundstream : An end-to-end neural audio codec. *CoRR*, 2021.