Codage audio avec une modélisation de spectrogramme par auto-encodeur

Mohamed Yaoumi, Pierre Mahe, Stéphane Ragot

Codage audio avec une modélisation de spectrogramme par auto-encodeur

Mohamed YAOUMI*, Pierre MAHÉ*, Stéphane RAGOT

Orange Labs, 2 Avenue Pierre Marzin, 22300 Lannion, France
med.yaoumi@gmail.com, mahe.pierre@live.com, stephane.ragot@orange.com

Résumé – Cet article explore une façon d'adapter à l'audio les méthodes récentes de compression d'images et vidéo par auto-encodeur. Le signal audio est analysé par transformée MDCT sur des trames successives; le spectrogramme d'amplitude, vu comme une image 2D, est compressé par auto-encodeur, les signes sont codés à part. Les performances pour la parole pleine bande sont comparées à des codecs traditionnels (MP3, Opus); l'évaluation objective de qualité (segSNR, PEAQ) est complétée par des écoutes expertes informelles.

Abstract – This article explores how recent image and video coding methods based on autoencoders may be applied to audio. The input audio signal is analyzed by MDCT transform over successive time frames; the magnitude spectrogram is compressed as a 2D image using an auto-encoder, while signs are coded separately. The coding performance for fullband speech is compared with traditional codecs (MP3, Opus). Test results are presented in terms of objective quality evaluation (segSNR, PEAQ) and informal expert listening.

1 Introduction

On considère ici le problème du codage audio (mono). Les méthodes fondamentales de codage avec perte, comme le codage par prédiction linéaire (LPC) ou par transformée (typiquement une transformée en cosinus discrète modifiée ou MDCT), visent à transformer le signal d'entrée en une représentation plus compacte, donnant un gain de codage [1]. Les codecs avancés, tels que EVS [2] et Opus [3] en voix sur IP, ou MPEG USAC [4] en diffusion ou stockage, étendent ces principes en combinant de nombreux blocs de traitement pour l'analyse/synthèse, la quantification ou le codage entropique. L'optimisation globale de tels sytèmes de codage traditionnels est complexe ; le codage peut se faire de bout en bout par analyse par synthèse, mais ce principe est en général limité à la recherche optimale d'un sous-ensemble de paramètres de codage. Récemment, de nouvelles approches de compression de signaux par réseaux de neurones ont émergé. En particulier, la compression par auto-encodeur [5–7] atteint des performances proches des codecs de l'état de l'art en codage d'images et vidéo [8, 9]. Les réseaux de neurones permettent d'apprendre des transformations non linéaires potentiellement plus efficaces que les représentations linéaires traditionnelles.

Pour le codage audio, les réseaux de neurones peut être utilisés de plusieurs manières. La première manière consiste à analyser le signal d'entrée afin de piloter le mode de codage. Par exemple, le codec Opus [3] a intégré une classification parole/musique à base d'un réseau de neurones récurrent (RNN). Une seconde manière consiste à générer ou corriger directement un signal audio. Dans [10], un réseau antagoniste génératif (GAN) sert à rehausser la qualité audio du signal codé par un codec mono traditionnel. Dans [11], les hautes fréquences sont synthétisées à l'aide d'un auto-encodeur. Pour la synthèse vocale, le modèle *Wavenet* [12] s'est imposé comme un modèle de référence. La particularité de ce modèle récursif est de générer le signal échantillon par échantillon. Dans [13], une analyse LPC est effectuée sur le signal d'entrée pour contrôler le modèle de synthèse vocale ; contrairement au codage LPC classique, le résidu de prédiction est généré par un modèle *WaveRNN* qui est une variante du modèle *Wavenet*. Ainsi, le codage de parole par réseau de neurones atteint des performances supérieures à l'état de l'art à très bas débit [13].

On s'intéresse ici au codage audio par auto-encodeur. Le codage VQ-VAE [14] combine un auto-encodeur variationnel (VAE) et la quantification vectorielle (VQ), avec des applications à la conversion de locuteur ou la modélisation pour la synthèse de parole. Cette approche a été améliorée, avec le codec SoundStream [15] basé sur un auto-encodeur et une quantification vectorielle résiduelle ; ce modèle donne d'excellents résultats à très bas débit pour le codage de parole en bande élargie ; un mécanisme de réduction de bruit ambiant peut être intégré lors du codage. La question étudiée ici est d'explorer comment les méthodes de codage d'images et vidéo par auto-encodeur [5, 6] peuvent être transposées à l'audio. Cet article est une prolongation des travaux réalisés dans [16, Chap. 7].

Cet article est structuré comme suit. Les modèles de codage par auto-encodeur de [5] ("modèle latent") et [6, 7] ("modèle hyper-latent") sont résumés à la section 2. Ces méthodes sont adaptées à l'audio à la section 3, avant de discuter des résultats expérimentaux à la section 4, et conclure à la section 5.

*M. Yaoumi et P. Mahé étaient à Orange Labs quand ces travaux ont été réalisés. Ces travaux ont été initiés dans un chapitre de la thèse de P. Mahé (Université de La Rochelle sous la direction de Sylvain Marchand).

2 Compression par auto-encodeur

Les auto-encodeurs sont des algorithmes d'apprentissage par réseaux de neurones artificiels qui permettent de construire une nouvelle représentation plus compacte d'un jeu de données. Un auto-encodeur est constitué de deux parties : l'encodeur (partie analyse) qui transforme le signal d'entrée en un espace latent de dimension plus faible et qui représente les caractéristiques importantes du signal d'entrée, et le décodeur (partie synthèse) qui reconstruit le signal à partir de l'espace latent.

La figure 1 illustre le modèle de [5] appelé "modèle latent" par la suite. Le signal d'entrée $\mathbf{x}$ est transformé en une représentation latente $\mathbf{y} = g_a(\mathbf{x}; \theta_{g_a})$, où g_a est la partie analyse de l'auto-encodeur et θ_{g_a} correspond aux paramètres d'analyse. Ensuite, $\mathbf{y}$ est quantifiée. A la phase d'inférence cette opération correspond à : $\hat{\mathbf{y}} = \lfloor \mathbf{y} \rceil$ où $\lfloor . \rceil$ est l'arrondi à l'entier le plus proche. Puis, $\hat{\mathbf{y}}$ est codé par un codeur entropique. Lors de l'apprentissage, cette opération quantification est remplacée par une approximation [5] : $\tilde{y} = \mathbf{y} + \Delta$ où $\Delta \sim \mathcal{U}[-\frac{1}{2}; \frac{1}{2}]$ a une distribution uniforme i.i.d. Après décodage entropique, le signal est reconstruit comme $\hat{\mathbf{x}} = g_s(\hat{\mathbf{y}}, \theta_{g_s})$, où g_s est la partie synthèse de l'auto-encodeur et θ_{g_s} correspond aux paramètres de synthèse. L'auto-encodeur est réalisé dans [5] par des couches de convolution avec N filtres, donnant N cartes d'activation (*feature maps*) latentes.

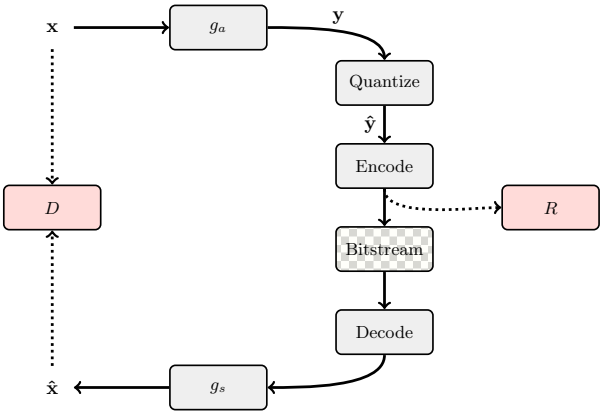


FIGURE 1 – Compression par auto-encodeur.

A la phase d'apprentissage, les paramètres θ_{g_a} et θ_{g_s} peuvent être optimisés pour la fonction de coût de la forme :

$$\mathcal{L}(\lambda) = R(\hat{\mathbf{y}}) + \lambda D(\mathbf{x}, \hat{\mathbf{x}}) \quad (1)$$

où D est la distorsion moyenne et R est le débit moyen. Le compromis débit/distorsion est paramétrable par λ. La distorsion correspond ici à l'erreur quadratique moyenne (MSE) entre $\mathbf{x}$ et $\hat{\mathbf{x}}$. Dans [5], deux hypothèses sont faites sur l'espace latent $\mathbf{y}$: les valeurs de chaque carte d'activation $\mathbf{y}_i$ sont supposées indépendantes d'une carte à l'autre ; à l'intérieur de la $i^{\text{ième}}$ carte latente, tous les éléments sont supposés suivre une même distribution $p_{\mathbf{y}_i}$, ce qui conduit à :

$$p_{\mathbf{y}} = \prod_{i,j,k} p_{\mathbf{y}_i}(\mathbf{y}_{i,j,k}) \quad (2)$$

où j et k sont les coordonnées d'un élément de la $i^{\text{ième}}$ carte d'activation et $p_{\mathbf{y}_i}$ est modélisée par une distribution gaussienne $p_{\mathbf{y}_i} = \mathcal{N}(0, \sigma_i^2)$, où σ_i^2 est estimée lors de l'apprentissage.

Dans [6, 7], le modèle de compression est étendu en ajoutant un réseau de neurones supplémentaire pour représenter l'espace latent $\mathbf{y}$. On définit ainsi l'espace hyper-latent $\mathbf{z} = h_a(\mathbf{y}; \theta_{h_a})$, où h_a est la partie analyse, incluant des couches de convolution avec M filtres. L'espace hyper-latent $\mathbf{z}$ est quantifié par arrondi (ou approximé par ajout de bruit) pour obtenir $\hat{\mathbf{z}}$ (ou $\tilde{\mathbf{z}}$ à l'apprentissage), avant codage entropique ; après décodage de $\hat{\mathbf{z}}$ (ou $\tilde{\mathbf{z}}$), on reconstruit $\Upsilon = h_s(\hat{\mathbf{z}}; \theta_{h_s})$, où h_s est la partie synthèse de l'auto-encodeur. Des détails sur cette extension, appelée "modèle hyper-latent" par la suite, sont donnés dans [16, Chap. 7]. L'espace hyper-latent sert à modéliser la distribution de chaque élément de l'espace latent par $p_{\mathbf{y}_{i,j,k}} = \mathcal{N}(\hat{\mu}_{i,j,k}, \hat{\sigma}^2_{i,j,k})$, où $\hat{\mu}_{i,j,k}$ et $\hat{\sigma}^2_{i,j,k}$ sont déterminés à partir de Υ [7]. Le codage entropique de $\hat{\mathbf{y}}$ du modèle latent est ainsi amélioré en exploitant ce modèle de distribution. Le débit $R(\hat{\mathbf{y}})$ dans $\mathcal{L}(\lambda)$ est remplacé par $R(\hat{\mathbf{y}}) + R(\hat{\mathbf{z}})$.

3 Adaptations à l'audio

A la différence des signaux d'images ou vidéo, un signal audio est signé et possède une dynamique importante (supposée sur 16 bits). On propose donc d'adapter la compression par auto-encodeur décrite à la section 2 de la façon suivante :

— Le signal audio est transformé en spectrogramme codé comme une image 2D ; le modèle par auto-encodeur est appliqué aux amplitudes, les signes sont codés à part.

— Une compression non linéaire est appliquée aux amplitudes. Une telle compression a été par exemple proposée dans [17] pour réduire la dynamique d'une transformée de Fourier en entrée d'un modèle.

Dans la suite, les signaux audio $\mathbf{s}$ sont supposés échantillonnés à une cadence f_s. Lors de l'apprentissage, le codec prend en entrée un segment de N_T trames de L échantillons extraits de chaque signal de la base d'apprentissage. On considère ici $f_s = 48$ kHz, $L = 128$ et $N_T = 375$ (soit 1 seconde de signal), avec une découpe positionnée aléatoirement pour chaque signal. Lors de l'inférence, le codec prend en entrée toutes les trames de L échantillons du signal. On suppose donc un scénario sans contrainte de retard pour des applications de stockage.

Chaque trame (d'indice i) du signal $\mathbf{s}(i, n), n = 0, \cdots, L-1$ est analysée par transformée MDCT (avec un recouvrement de 50% et un fenêtrage sinusoïdal sur $2L$ échantillons) pour obtenir le spectre $S(i, k), k = 0, \cdots, L-1$. La MDCT a été choisie pour deux raisons : fournir un spectrogramme de dimensions similaires à celles utilisées dans un modèle de compression d'images ; permettre une intégration plus facile dans une architecture de codage audio traditionnel. Le spectre $|S(i, k)|$ est normalisé sur $[0, 1]$. Le facteur de normalisation doit être transmis comme métadonnée au décodeur. Le débit associé étant faible (< 16 bits), il est négligé ici. La transformée MDCT

normalisée est compressée selon la loi μ [1] où $\mu = 255$:

$$A(i,k) = \frac{ln(1+\mu\frac{|S(i,k)|}{\max_{i,k}|S(i,k)|})}{ln(1+\mu)}. \quad (3)$$

La figure 2 montre l'architecture de l'auto-encodeur dans le "modèle latent", où l'entrée $\mathbf{x}$ correspond au spectrogramme 2D donné par $A(i,k)$. La partie analyse g_a est composée de quatre couches, constituées d'une convolution 2D avec des filtres 5×5 suivie d'une décimation par 2. Après les 3 premières couches, une fonction d'activation de type *Leaky ReLU* est utilisée. La dernière couche n'a aucune activation pour ne pas limiter les valeurs que peut prendre la sortie $\mathbf{y}$, sachant que la distribution $p_{\mathbf{y}_i}$ est supposée suivre une loi normale. A la différence de [5], aucune normalisation de type GDN (pour *generalized divisive normalization*) n'est utilisée entre chaque couche. Dans la phase d'inférence, les valeurs entières de l'espace latent sont ensuite codées par un codeur arithmétique (AC) puis le train binaire est transmis vers le décodeur.

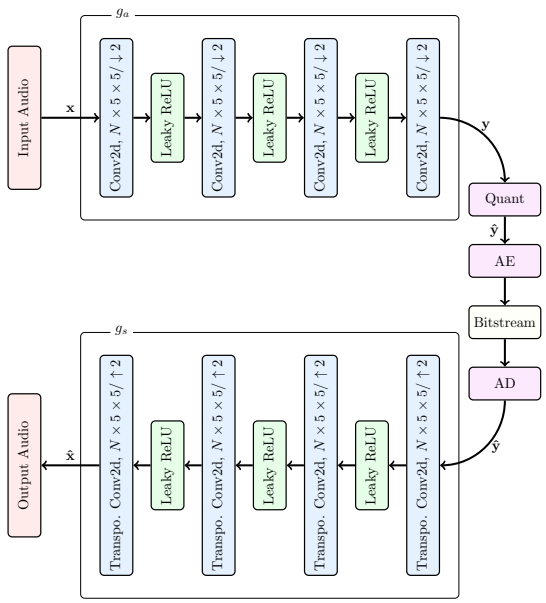


FIGURE 2 – Architecture (modèle latent).

Au décodeur, on obtient $\hat{\mathbf{y}}$ ou $\tilde{\mathbf{y}}$. La partie synthèse g_s a une architecture construite en miroir par rapport à la partie analyse g_a. Elle comprend 4 couches successives de convolution transposée 2D. Dans l'architecture originale [5], une convolution 2D et une interpolation linéaire sont utilisées ; la convolution transposée permet une interpolation non linéaire plus riche qu'une simple pondération linéaire des valeurs [7]. Le nombre de cartes d'activation N permet de donner plus ou moins de degrés de liberté au modèle pour représenter les signaux d'entrée.

Les paramètres θ_{g_a} et θ_{g_s} du réseau sont appris en utilisant l'algorithme d'optimisation Adam. Le taux d'apprentissage (*learning rate*) est initialisé à $\alpha = 10^{-4}$, il est divisé par 5 tous les 100 époques (*epoch*) jusqu'à atteindre $\alpha = 8 \times 10^{-7}$. Chaque apprentissage est réalisé sur un ensemble de 300 époques. La taille du *batch* est fixée à 8. Lors de la phase d'initialisation, les différentes cartes d'activation sont initialisées selon l'initialisation de Glorot.

Dans le modèle hyper-latent, l'auto-encodeur de la figure 2 est complété par un réseau de neurones supplémentaire détaillé dans [16, Chap. 7]. Il comprend 3 couches de convolution 2D avec $M = 64$ cartes d'activation. La première a des filtres 3×3 sans décimation par 2 et une activation *Leaky ReLU* ; les deux autres couches ont des filtres 5×5 avec une décimation par 2 et une activation *Leaky ReLU* uniquement dans la deuxième couche.

Après la partie synthèse de l'auto-encodeur, le spectrogramme 2D d'amplitude $\hat{\mathbf{x}}$ est post-traité avec une expansion inversant la compression selon la loi μ et une dénormalisation. Les signes (1 bit par raie MDCT utile) sont démultiplexés et appliqués avant la transformée MDCT inverse.

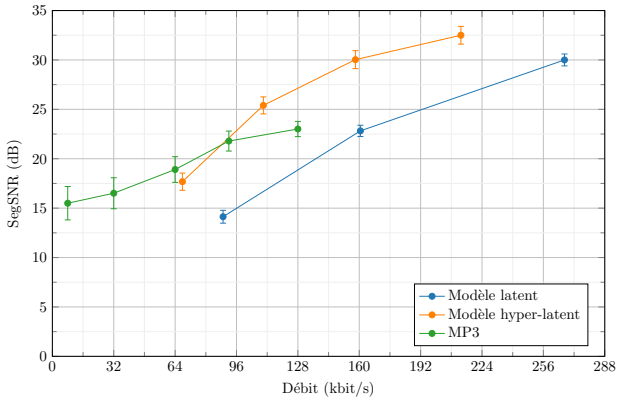
4 Résultats expérimentaux

Les performances de codage sont évaluées pour des signaux de parole pleine bande échantillonnés à $f_s = 48$ kHz, avec deux codecs de référence Opus (v.1.3.1) et MP3 (lame v3.1) utilisés ici à débit fixe – respectivement 16, 24, 32, 64, 96 kbit/s et 32, 48, 64, 92, 128 kbit/s. Pour des raisons de reproductibilité et cette étude étant exploratoire, la qualité est principalement évaluée selon deux métriques objectives : rapport signal sur bruit segmental (segSNR) – avec des segments de 20 ms – et PEAQ [18] – PEAQ est ici en version "basic", selon l'implémentation de P. Kabal (McGill Univ.).

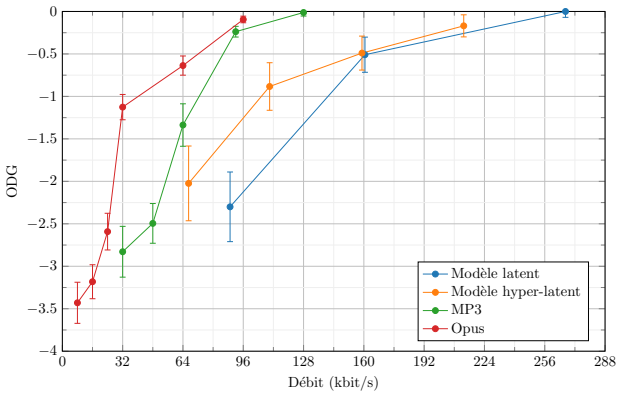
L'apprentissage et l'évaluation ont été faits sur le corpus VTCK [19]. Ce corpus est composé d'enregistrements de 400 phrases en anglais prononcées par 109 locuteurs, pour un total d'environ 40000 extraits audio. La durée des phrases est variable – comprise entre 2 et 20 secondes. Les segments de silence ont été supprimés des échantillons d'origine pour éviter un biais lors de l'apprentissage. Les échantillons ont été normalisés à -23 dBov (RMS). Le corpus a été divisé en 3 parties, donnant les bases d'entraînement, de validation et de test – les échantillons ont été répartis aléatoirement selon la proportion de 80%, 10%, 10% respectivement.

Les apprentissages ont été mis en œuvre sur des machines équipées d'une carte graphique *Nvidia GeForce RTX 2080 Ti* et d'un processeur *Intel Core i9-9940X*. La réalisation est en *Python* avec le *framework Pytorch*. Le paramètre λ de la fonction de coût est fixé respectivement à $\lambda = 10^2, 10^3, 10^4$ et $\lambda = 10^2, 10^3, 4.10^3, 10^4$ pour les modèles latent et hyper-latent.

Les scores segSNR et ODG (*Overall Degradation Grade*) [18] moyens et les débits moyens sont présentés à la figure 3, avec les écarts-types. Le modèle hyper-latent permet un codage à plus bas débit que le modèle latent. En termes de segSNR, le modèle hyper-latent est équivalent ou meilleur que MP3 – Opus n'est pas évalué car ce codec intègre des non-linéarités qui rendent le critère segSNR non pertinent. Les scores PEAQ du modèle hyper-latent sont inférieurs à ceux d'Opus et MP3.



(a) segSNR



(b) PEAQ

FIGURE 3 – Qualité objective sur la parole pleine bande.

A noter que le codage de signes dans la méthode proposée requiert 40 kbit/s (1 bit par raie MDCT pour la bande utile 0–20 kHz) ; il serait possible de réduire ce débit "plancher" en ne codant en hautes fréquences que les signes les plus importants.

Des écoutes expertes ont montré que la chute de qualité à bas débit pour Opus et MP3 est due à une limitation de bande, bien qu'il y ait aussi des artéfacts de codage dépendant du codec. Pour les modèles latent et hyper-latent la bande audio est conservée à tous les débits testés, mais un bruit modulé (dont le niveau dépend du signal d'entrée et du débit) est audible aux plus bas débits ; l'écart-type des scores ODG est plus large, la qualité audio est plus variable que pour Opus et MP3. Un test subjectif formel est prévu pour confirmer ces observations.

5 Conclusion

Cet article a présenté une nouvelle approche de codage audio par transformée où le spectogramme d'amplitude (MDCT) est représenté par un auto-encodeur. Les résultats expérimentaux montrent que le compromis débit/distorsion n'est pas encore compétitif par rapport à un codec de l'état de l'art comme Opus.

La compression audio par réseau de neurones est un champ de recherche très récent et de nombreuses pistes restent à explorer, en dehors de l'optimisation complète de l'architecture et des hyper-paramètres (par exemple longueur de trames L, normalisation des couches ...). Il serait intéressant d'intégrer les aspects perceptifs, dans la fonction de coût ou en utilisant un pré/post-traitement de pondération perceptuelle. De plus, l'utilisation d'une transformée entraine des limitations [17]. Un auto-encodeur dans le domaine temporel comme [15] semble plus adapté, une comparaison directe avec [15] reste à réaliser. Ces travaux doivent être étendus à l'audio général (dont parole bruitée, musique, contenu mixte). Le codage pourrait être adapté à des contraintes de faible latence, avec un traitement temps-réel par trames de typiquement 20 ms.

Références

[1] N. S. Jayant and P. Noll, *Digital Coding of Waveforms : Principles and Applications to Speech and Video*. Prentice-Hall, 1984.

[2] M. Dietz *et al.*, "Overview of the EVS codec architecture," in *Proc. ICASSP*, 2015.

[3] J.-M. Valin *et al.*, "High-quality, low-delay music coding in the opus codec," *Journal of the Audio Engineering Society*, 2016.

[4] M. Neuendorff *et al.*, "The ISO/MPEG Unified Speech and Audio Coding Standard – Consistent High Quality for all Content Types and at all Bit Rates," *JAES*, 2013.

[5] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *Proc. ICLR*, 2017.

[6] J. Ballé *et al.*, "Variational image compression with a scale hyperprior," in *Proc. ICLR*, 2018.

[7] D. Minnen *et al.*, "Joint autoregressive and hierarchical priors for learned image compression," in *Proc. NIPS*, 2018.

[8] T. Ladune *et al.*, "Binary probability model for learning based image compression," in *Proc. ICASSP*, 2020.

[9] T. Ladune *et al.*, "Conditional coding for flexible learned video compression," *Proc. ICLR*, 2021.

[10] A. Biswas and D. Jia, "Audio codec enhancement with generative adversarial networks," in *Proc. ICASSP*, pp. 356–360, 2020.

[11] M. Miron and M. Davies, "High frequency magnitude spectrogram reconstruction for music mixtures using convolutional autoencoders," in *Proc. DAFx*, pp. 173–180, 2018.

[12] A. van den Oord *et al.*, "WaveNet : A generative model for raw audio," in *ISCA Speech Synthesis Workshop*, 2016.

[13] J. Skoglund and J.-M. Valin, "Improving Opus low bit rate quality with neural speech synthesis," *Proc. Interspeech*, 2020.

[14] A. van den Oord *et al.*, "Neural Discrete Representation Learning," in *Proc. NIPS*, 2017.

[15] N. Zeghidour *et al.*, "Soundstream : An end-to-end neural audio codec," *IEEE/ACM Trans. ASLP*, vol. 30, pp. 495—-507, 2022.

[16] P. Mahé, *Codage Ambisonique pour les Communications immersives*. PhD thesis, Université de La Rochelle, France, Feb. 2022.

[17] H. Caracalla and A. Roebel, "Sound texture synthesis using RI spectrograms," in *Proc. ICASSP*, pp. 416–420, 2020.

[18] T. Thiede *et al.*, "PEAQ - The ITU standard for objective measurement of perceived audio quality," *Journal of the Audio Engineering Society*, vol. 48, no. 1/2, pp. 3–29, 2000.

[19] J. Yamagishi *et al.*, "CSTR VCTK Corpus : English multi-speaker corpus for CSTR voice cloning toolkit," 2019.