UNIVERSITÉ DE LORRAINE | BIBLIOTHÈQUES UNIVERSITAIRES

AVERTISSEMENT
DROIT D'AUTEUR

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document. Toute contrefaçon, plagiat ou reproduction illicite encourt une poursuite pénale.

Contact bibliothèque : ddoc-theses-contact@univ-lorraine.fr
(Cette adresse ne permet pas de contacter les auteurs)

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10

UNIVERSITÉ DE LORRAINE
34 COURS LÉOPOLD — CS 25233
54052 NANCY CEDEX
TÉL : 03 54 50 54 00
CONTACT@UNIV-LORRAINE.FR
WWW.UNIV-LORRAINE.FR

Ecole Doctorale: IAEM
Laboratoire: Loria

Thèse

Présentée et soutenue publiquement pour l'obtention du titre de

DOCTEUR DE l'UNIVERSITE DE LORRAINE

Mention : INFORMATIQUE

par **Robin San Roman**

Sous la direction de Romain Serizel et Antoine Deleforge

Safe Audio Generation Models from Discrete Latent Representations

Soutenance le 15/12/2025

Membres du jury :

Directeurs de thèse :	**M. Romain Serizel**	**Full professor - HDR, LORIA Univ. de Lorraine, Nancy**
	M. Antoine Deleforge	**Tenured research scientist at Inria, Strasbourg**
Président de jury :	**Mme. Armelle Brun**	**Full professor - HDR, LORIA, Univ. De Lorraine, Nancy**
Rapporteurs :	**M. Timo Gerkmann**	**Full professor, Universität Hamburg, Hambourg**
	M. Wenwu Wang	**Full professor, University of Surrey, Guildford UK**
Examinateurs :	**Mme. Karen Livescu**	**Full professor, Toyota Technological Institute, Chicago**
	M. Simon Leglaive	**Maître de Conférences à CentraleSupélec Paris**
Membres invités :	**M. Yossi Adi**	**Associate Professor, Meta and the Hebrew University of Jerusalem, Jerusalem**

Table of Contents

Acknowledgments **5**

General Introduction **22**

General Bibliography **24**

1 Digital audio processing **24**
1.1 Fourier transform 24
1.2 Short-time Fourier transform (STFT) 25
1.3 Audio Compression 28

2 Machine learning basics **29**
2.1 Definitions 29
2.2 Supervised learning 30
2.3 Unsupervised learning 30

3 Audio Representation Learning **32**
3.1 Supervised learning 32
3.2 Self Supervised Learning 33
3.3 Discrete Audio Representations 34

4 Audio generative Models **38**
4.1 Vocoders 38

4.2 Creative models . . . 40
4.3 Evaluation . . . 42

I Generative Audio Models from Discrete Representations 46

5 Introduction to Audio Language Models 47
5.1 Motivations . . . 47
5.2 Neural Audio Codecs . . . 49
5.3 Audio Language Modeling . . . 53
5.4 Applications . . . 57
5.5 Concurrent Approaches . . . 59

6 Decoding Audio Tokens with Diffusion 61
6.1 Introduction . . . 61
6.2 Method . . . 64
6.3 Experimental Setup . . . 68
6.4 Results . . . 71
6.5 Conclusion . . . 79

7 Stem based Music Generation 80
7.1 Introduction . . . 80
7.2 Related Work . . . 82
7.3 Method . . . 84
7.4 Experiments . . . 86
7.5 Conclusion . . . 89

II Watermarking for Safer Audio Generative Modeling 90

8 Introduction to Audio Watermarking 91
8.1 Challenges . . . 92
8.2 Watermarking for Generative AI content . . . 95
8.3 Traditional Watermarking Methods . . . 96
8.4 Deep Learning Watermarking Methods . . . 97

9 Improving Audio Watermarking Models (Audioseal) 99

9.1 Introduction . . . 99
9.2 Method . . . 102
9.3 Audio/Speech Quality . . . 107
9.4 Experiments and Evaluation . . . 108
9.5 Adversarial Watermark Removal . . . 114
9.6 Conclusion . . . 116
9.7 False Positive Rates - Theory and Practice . . . 118
9.8 Additional Experimental Results . . . 119
9.9 Experimental details . . . 123

10 Safe and Open-Sourceable Audio LMs with Latent Watermarking 127
10.1 Introduction . . . 127
10.2 Audio Model Watermarking . . . 129
10.3 Experiments . . . 131
10.4 Attacks on the Model's Watermark . . . 134
10.5 Conclusion & Discussion . . . 136

11 Conclusion 137

Bibliography 141

Acknowledgments

My PhD years have been an incredible time, probably the most intellectually stimulating years of my life. Some days it felt like I was stagnating, but looking back, the progress was constant; it just doesn't show in real time. The only scar I won't recover from is that I can no longer listen to my test song *Californication*. None of this would have happened without the people around me, and this page is for them.

Thank you to my Meta manager Yossi Adi, for his support, his positive mood, and his kindness throughout these three years. And for the groove lesson in the Paris music room! I also want to thank Alexandre Défossez and Gabriel Synnaeve for their guidance, especially at the beginning of the PhD: their help was crucial to get over the hump of getting my first conference paper out, and they also introduced me to Antoine and Romain.

Huge thanks to my two brilliant advisors Romain Serizel and Antoine Deleforge for being consistently there every week, with sharp feedback and a lot of patience. Unfortunately we didn't solve audio once and for all through our phase problem, but I am sure that time will come! Thank you both for being the best advisors even from 400 km away.

Thanks also to Simon R., Teddy F., Hady E., Axel R., Tuan T., and Pierre F. for the collaborations and the discussions that shaped much of this work. Very special thanks to Pierre Fernandez for onboarding me into the wonderful world of watermarking, which ended up being a significant part of my PhD. And to Simon Rouard, for all the time spent on audio codecs and on figuring things out together, while both lost and lonely in our PhD journey. Previous research internship mentors: E. Dupoux, Paul Michel, S.

Acheche, T. Courtat, J. Carmona. Thank you for showing me what research looked like before I jumped in. Eliya N. and Lior W., I will always be grateful that you allowed me to have this rare opportunity, and to meet all the people I ended up working with during this PhD.

Thank you to all the Meta PhDs, postdocs, and other researchers who have been a fantastic team to live with during these three years. Adrien, Mathurin, and Wassim, aka the cool kids of the back of the office. The blue bag gang: Timothée, Quentin, Bard, Fabian, Pierre, Theo. Joao, always being there when I needed a drink, Tom, Fede, Mégi, Pierre, Belen, Kruno, Kunhao, Guillaume, Paul-Ambroise, Juliette, Jarod, Josephine, Ismail, Marlène, Mohamed, Marvine, Basile, Robin, Tu anh, Yongyu, Arjun, Théophane, Tariq, Onur, Matthijs, Jade, Loïc. Thank you to all the meta workers that made the life in the office so nice, security, culinary team, facility team and the CSE for the nice parties.

Thank you Anaëlle for your care and your patience, especially for putting up with me bringing stress and laptop work into a few holidays, and somehow never really making me feel bad about it. Thank you Rachid, Julia, Ben, Cec, Guillaume, Isa, and of course Tom, for helping me enjoy my much-needed vacations during these years. Thank you to Louise for making our apartment such a nice and lively place to live in.

My friends from prépa: Ecoffet, Fabiani, Mika Rey, Masson, Gerard, Maude, Sintes, Kenza, Leati, Popi, Sophie, Vilatte, Lily, Marion, Marius, Paul K., Valbuena, Quentin, Gaspard, Bapt, Céline, Périne. S/O to my ENS crew: Alain, Léonard, Thibaud, Marius (him again), Nathan, Hichem. Thank you Adele, for the years of support during prépa/undergrad that helped me get to this point. Thank you to all my lanceurs de bâton du Benou Reiter, Will, Gautier, Liane, Ines, Isaure, and Alicia. Thank you Koffio for all the 1-1 on Naruto after work. Shout out to Ilym's members Florian, Ben, Guillaume and Guillaume. And others who matter: JP (even if it's hard to get to this guy), Corentin, Margaux, Victoire, Ronteau, Théo, Fanny, Ugo, Albert, Celia, Lucio.

Thank you to my incredible parents for your care and the education you gave me. I appreciate you pushing me all along, but never too hard. Last but not least thank you to the two goats, the big bosses of the universe Flo and Sam.

And finally, a thought for Pierrelee, dear friend gone too soon, but never forgotten. This one's for you.

Résumé court

Le développement rapide du deep learning a profondément transformé le traitement du signal audio, permettant aux modèles d'apprendre directement à partir du signal brut et d'atteindre des niveaux de réalisme inédits en synthèse vocale comme en génération musicale. Cette avancée s'accompagne toutefois de nouveaux risques : à mesure que les contenus générés deviennent indiscernables des enregistrements réels, des enjeux majeurs émergent autour de la désinformation, de l'usurpation d'identité et du respect du droit d'auteur. La question centrale devient alors celle d'une génération audio à la fois performante et responsable.

Dans ce contexte, la thèse s'articule autour de deux axes principaux : l'amélioration des modèles génératifs audio fondés sur des représentations latentes discrètes, et l'intégration de techniques de tatouage numérique pour garantir la traçabilité et la sécurité des contenus générés.

Côté génération, cette thèse se concentre sur les modèles de langage audio, qui adaptent les principes des modèles de langage textuel à l'audio via des séquences de tokens issues de codecs neuronaux. Une méthode de décodage par diffusion multi-bande est introduite, offrant une amélioration notable de la fidélité sonore par rapport aux vocodeurs classiques, aussi bien pour la parole que pour la musique. Par ailleurs, une approche multi-pistes pour la génération musicale permet de produire et d'éditer séparément différentes pistes instrumentales, rendant possible une édition musicale plus fine et adaptée aux pratiques créatives.

Sur le plan de la sécurité, la thèse propose AudioSeal, une méthode de tatouage audio

localisé, robuste et imperceptible, capable de détecter la présence d'un watermark même après de nombreuses altérations du signal. Cette approche permet une détection rapide et localisée, adaptée à des usages à grande échelle. En outre, une méthode de tatouage intra-modèle est développée pour les modèles open source. En rendant le watermark résistant à la compression neuronale, il devient possible d'assurer la traçabilité des contenus générés via tatouage des données d'entrainement. Avec cette stratégie, les sorties du modèle sont détectables même en cas de modification du modèle ou du décodeur.

En synthèse, la thèse pose les bases d'une génération audio à la fois de haute qualité, contrôlable et traçable, conciliant innovation technologique et exigences éthiques, et ouvrant la voie à des usages créatifs et responsables de l'IA audio.

Résumé étendu

Contexte et motivations

Au cours de la dernière décennie, les progrès de l'apprentissage profond ont profondément remodelé le traitement automatique des signaux audio. Avant cette mutation, les pipelines de synthèse et d'analyse reposaient essentiellement sur des chaînes de traitement du signal soigneusement conçues à la main : transformées de Fourier ou par ondelettes pour l'extraction de caractéristiques, modèles de mélanges gaussiens et modèles de Markov cachés pour la reconnaissance vocale, synthèse concaténative pour la lecture de texte, et approches symboliques ou fondées sur des règles pour la génération musicale. Si ces méthodes étaient bien comprises et efficaces sur le plan calculatoire, elles dépendaient fortement de descripteurs conçus manuellement et d'heuristiques spécifiques à chaque tâche, ce qui limitait leur capacité de généralisation et la qualité perceptive de leurs sorties.

L'arrivée des réseaux de neurones profonds, d'abord pour la reconnaissance vocale supervisée, puis pour les tâches génératives, a fait basculer ce paysage. Les modèles ont commencé à apprendre directement à partir du signal brut ou peu transformé, en découvrant des représentations optimisées pour la tâche à résoudre plutôt qu'en se reposant sur des descripteurs choisis a priori par l'humain. Les avancées en apprentissage de représentations, en modélisation de séquences et en codecs neuronaux ont permis aux modèles de capturer des dépendances temporelles longues et des structures acoustiques fines avec une précision inédite. Cette dynamique a donné naissance à des systèmes de synthèse vocale capables de produire des voix d'un naturel saisissant, à des outils de génération musicale capables de composer des morceaux entiers à partir d'une simple description textuelle, et

à des modèles audio généraux capables d'opérer sur plusieurs modalités.

Cependant, ce qui fait la puissance de ces modèles soulève également des défis majeurs. À mesure que la qualité des sorties s'améliore, les signaux générés deviennent de plus en plus difficiles à distinguer d'enregistrements authentiques, ce qui entraîne des risques importants en matière de désinformation, d'usurpation d'identité, et de respect du droit d'auteur. Cette dualité a réorienté une partie de la communauté scientifique : la question n'est plus seulement *peut-on générer du son de haute qualité ?*, mais bien *comment le faire de manière sûre et responsable ?*. Parmi les pistes proposées, le tatouage numérique (*watermarking*) s'est imposé comme une approche prometteuse pour identifier ou tracer de manière fiable les contenus audio générés par IA, même après altérations utilisateur ou traitements des plateformes.

Dans ce paysage, cette thèse explore deux axes complémentaires. Le premier porte sur l'amélioration des modèles génératifs audio fondés sur des représentations latentes discrètes, en particulier les *modèles de langage audio* (Audio Language Models, ou ALMs), avec un double objectif : améliorer la qualité acoustique de leurs sorties, et augmenter leur contrôlabilité par l'utilisateur. Le second axe concerne le déploiement responsable de ces modèles, en intégrant des techniques de tatouage numérique conçues pour s'inscrire dans le pipeline génératif sans dégrader la qualité, tout en restant compatibles avec les contraintes des modèles open-source.

Les contributions présentées s'articulent autour de quatre articles publiés dans des conférences internationales : *Multi-Band Diffusion* (NeurIPS 2023), *MusicGen-Stem* (ICASSP 2025), *AudioSeal* (ICML 2024) et *Latent Watermarking of Audio Language Models* (ICASSP 2025).

Partie I – Modèles génératifs audio à partir de représentations discrètes

Modèles de langage audio et représentations discrètes

Les modèles de langage audio adaptent le paradigme des modèles de langage textuels au signal sonore. Le pipeline classique se décompose en trois étapes : (i) une *tokenisation* qui convertit le signal continu en une séquence discrète de tokens via un codec neuronal (par exemple EnCodec), (ii) un *modèle autorégressif* de type Transformer qui apprend la

distribution conditionnelle des tokens, et (iii) un *décodeur* qui reconstruit la forme d'onde à partir de la séquence de tokens prédite. Cette architecture en trois temps permet de tirer parti des progrès considérables réalisés en modélisation séquentielle dans le domaine du texte, tout en exploitant des représentations discrètes adaptées au signal audio. Cependant, chacun de ces étages introduit ses propres limitations : le codec neuronal peut être source d'artefacts à très bas débit, le modèle autorégressif est sujet aux dérives sur des séquences longues, et le décodeur GAN classique peine à restituer fidèlement les contenus complexes (musique, sons d'environnement).

Les deux premières contributions de la thèse adressent respectivement le décodage et la contrôlabilité de ce pipeline.

Multi-Band Diffusion : décoder les tokens audio par diffusion multi-bande

La première contribution, publiée à NeurIPS 2023, est une méthode de décodage par diffusion appelée **Multi-Band Diffusion** (MBD). Elle constitue une alternative haute fidélité aux décodeurs GAN traditionnels utilisés dans les codecs neuronaux comme EnCodec.

Le constat de départ est que les décodeurs GAN, bien que rapides, tendent à produire des artefacts perceptibles (voix métallique, distorsion, transitoires mal restitués) particulièrement à bas débit. Les modèles de diffusion, par leur formulation probabiliste, sont mieux à même de modéliser des distributions complexes, mais leur application directe à la forme d'onde audio se heurte à deux difficultés majeures. D'une part, le signal audio possède une structure fréquentielle riche, avec une énergie qui varie de manière non uniforme à travers les bandes : le bruit gaussien isotrope utilisé en diffusion classique ne s'accorde pas naturellement avec cette structure. D'autre part, les approches en cascade (un modèle de diffusion par bande, conditionné sur la sortie du précédent) sont coûteuses et difficiles à paralléliser.

Multi-Band Diffusion repose sur trois ingrédients principaux. D'abord, un *processus de diffusion spécifique à chaque bande de fréquence* : le signal est filtré en plusieurs bandes non chevauchantes, et chaque bande possède son propre modèle de diffusion. Cette décomposition réduit l'enchevêtrement des erreurs accumulées au fil des étapes de débruitage. Ensuite, un *égaliseur de fréquence (EQ processor)* réversible est appliqué en amont, qui

normalise l'énergie dans chaque bande afin de la rapprocher de la distribution gaussienne standard supposée par la formulation de la diffusion. Enfin, un *schéma de bruit de puissance* (*power noise scheduler*) est introduit, mieux adapté que le bruit gaussien classique aux signaux audio à fort contenu harmonique.

Le modèle est entraîné comme décodeur conditionnel : il prend en entrée les tokens discrets produits par EnCodec et reconstruit la forme d'onde correspondante. Les évaluations, à la fois objectives (métriques d'erreur spectrale, ViSQOL) et subjectives (études d'écoute avec annotateurs humains), montrent que MBD surpasse significativement les décodeurs GAN sur la parole, la musique, et les sons environnementaux. Le modèle s'intègre comme une option de décodage plus lente mais plus fidèle, sans nécessiter de modification de l'encodeur ni du quantificateur, ce qui en fait un remplacement *drop-in* de la partie décodeur d'EnCodec. MBD a ensuite été utilisé dans plusieurs travaux de génération audio, notamment dans MusicGen, comme décodeur alternatif pour les usages où la qualité prime sur la latence.

MusicGen-Stem : génération et édition multi-pistes par modèle autorégressif

La seconde contribution, publiée à ICASSP 2025, s'attaque au problème de la contrôlabilité des modèles de génération musicale. Les modèles de pointe comme MusicGen génèrent une piste audio unique à partir d'une description textuelle. Or, la pratique musicale réelle, en studio comme en composition, est fondamentalement *itérative* et *multi-pistes* : un producteur travaille piste par piste (basse, batterie, instruments mélodiques, voix), modifie chaque élément séparément, remplace une partie en gardant les autres intactes. Le texte, en tant que modalité de conditionnement, est trop grossier pour cette pratique : on ne sait pas décrire textuellement "la même batterie que dans ce morceau, mais avec un autre kit".

MusicGen-Stem est une extension de MusicGen qui adopte une approche multi-pistes : le modèle génère trois pistes en parallèle (*basse*, *batterie*, *autres*), chacune représentée par sa propre séquence de tokens. Plusieurs contributions techniques rendent cette approche viable.

Premièrement, la construction du jeu de données : la séparation en pistes est obtenue via un modèle de séparation de sources à l'état de l'art (Demucs). Cela permet de con-

stituer un grand corpus de musique segmentée, là où les jeux de données multi-pistes annotés à la main sont rares et limités en diversité. Les pistes vocales sont volontairement exclues, à la fois pour éviter la complexité de la génération cohérente de paroles et pour des raisons légales.

Deuxièmement, pour éviter les fuites entre pistes (*cross-talk*), des modèles de compression *spécialisés par piste* sont entraînés : un codec dédié à la basse, un autre à la batterie, un autre aux instruments restants. Cette spécialisation est essentielle, car un codec généraliste tend à encoder des informations résiduelles d'une piste dans une autre, ce qui empêche l'édition isolée.

Troisièmement, un schéma de conditionnement original permet à un même modèle de réaliser plusieurs tâches : génération à partir de texte de l'ensemble des trois pistes, génération conditionnelle d'une ou plusieurs pistes à partir des autres (édition piste-par-piste), et même modification de la texture d'une piste en régénérant les résidus de quantification vectorielle (RVQ) tout en gardant le premier flux fixe. Ce dernier point est particulièrement intéressant car il offre un mécanisme d'édition continue : la structure musicale (rythme, harmonie) est préservée, tandis que le timbre est modifié.

Les évaluations objectives et subjectives montrent que, malgré la complexité supplémentaire liée à la génération parallèle de trois pistes, le modèle reste au niveau de MusicGen sur la tâche de génération textuelle. Sur la génération inconditionnelle multi-pistes, il surpasse les modèles antérieurs ouverts (MSDM, Jen-1 Composer, StemGen). Sur la tâche d'édition de pistes, MusicGen-Stem se compare favorablement aux modèles open-source existants, en préservant beaucoup mieux les pistes non modifiées que les approches d'*instruction tuning* ou d'inversion par diffusion.

Cette contribution illustre un déplacement plus général dans la pensée des modèles génératifs audio : passer d'une génération "boîte noire" *tout-ou-rien*, à un paradigme structuré et centré sur l'utilisateur, où l'intention créative et le comportement du modèle sont mieux alignés.

Partie II – Tatouage numérique pour une génération audio plus sûre

Motivations et limites des méthodes existantes

À mesure que la qualité des modèles génératifs progresse, distinguer un enregistrement réel d'un contenu synthétisé devient impossible à l'oreille. Cela ouvre la voie à des usages malveillants : deepfakes vocaux à des fins de désinformation politique, hameçonnage par clonage de voix, contournement du droit d'auteur, ou détournement de la voix d'une personnalité publique. Les régulateurs ont commencé à exiger des mécanismes de transparence et de traçabilité (AI Act européen, ordre exécutif américain de 2023..).

Deux grandes familles de méthodes existent. La *détection passive* entraîne un classifieur binaire pour distinguer le réel du synthétique : approche simple, mais condamnée à perdre du terrain à mesure que les générateurs progressent. Le *tatouage numérique*, lui, est proactif : il consiste à insérer dans le signal généré une marque imperceptible mais détectable par un algorithme dédié. On distingue le tatouage *zéro-bit* (présence/absence) et le tatouage *multi-bit* (un message binaire identifiant un modèle ou un utilisateur).

Les méthodes existantes (WavMark, DEAR, etc.) souffrent de plusieurs limitations. D'abord, elles ne sont pas adaptées à la *détection* en pratique : leurs décodeurs n'ont jamais vu de signaux non tatoués durant l'entraînement, ce qui fausse les taux de fausse alarme. Ensuite, elles ne sont pas *localisées* : elles analysent l'intégralité d'un clip, ce qui rend difficile l'identification d'un court passage généré inséré dans un audio plus long. WavMark introduit une astuce de synchronisation par motifs répétés toutes les secondes, mais avec un coût élevé (31% de la capacité utile, détection par recherche exhaustive) qui la rend inadaptée à un usage temps réel sur de grandes plateformes.

AudioSeal : tatouage localisé robuste et imperceptible

La troisième contribution, publiée à ICML 2024, est **AudioSeal**, une méthode de tatouage audio *post-hoc* qui adresse les limites précédentes.

L'idée centrale est d'entraîner conjointement deux réseaux : un *générateur* qui prédit une forme d'onde additive constituant le tatouage à ajouter au signal hôte, et un *détecteur* qui produit, pour chaque échantillon de l'entrée, la probabilité qu'il soit tatoué. Là où les approches antérieures entraînent un décodeur qui tente de récupérer un message binaire

à partir de l'audio entier, AudioSeal abandonne cette logique et entraîne directement un *détecteur d'échantillon*. Cette différence est fondamentale : le détecteur produit une carte de probabilité à la résolution de l'échantillon (1/16000 seconde à 16 kHz), ce qui permet de *localiser* précisément les segments tatoués au sein d'un audio plus long.

Pour entraîner cette architecture à la localisation, on masque aléatoirement le tatouage sur des sections variables du signal pendant l'entraînement, et le détecteur doit apprendre à produire des sorties cohérentes avec ce masquage. Le générateur est entraîné conjointement pour maximiser la précision du détecteur tout en minimisant la différence perceptive entre signal original et signal tatoué. À cette fin, une *nouvelle fonction de cout perceptive* inspirée du masquage auditif est introduite : elle exploite les propriétés du système auditif humain (masquage fréquentiel, masquage temporel) pour permettre un tatouage plus énergétique dans les régions inaudibles, et donc plus robuste, tout en restant imperceptible.

AudioSeal est également étendu au tatouage *multi-bit* : un message binaire (par exemple, identifiant d'un modèle parmi plusieurs milliers) peut être inséré en plus du signal de détection, sans dégrader ce dernier. La robustesse est obtenue par augmentation pendant l'entraînement avec une large gamme d'attaques (compressions audio MP3 et Opus, ré-échantillonnage, filtres passe-bas/haut, bruit additif, edits temporels, etc.).

Les évaluations expérimentales montrent qu'AudioSeal atteint l'état de l'art sur tous les axes. En détection, il dépasse largement la détection passive et WavMark sur une vaste gamme d'éditions audio, atteignant des taux de détection proches de 100%. En localisation, il opère à la résolution de l'échantillon, là où WavMark se limite à 1 seconde. En vitesse, le détecteur est appelé une seule fois et fournit les logits de détection à chaque pas de temps, ce qui permet une détection *temps réel* sur des flux audio. Les gains en vitesse atteignent deux ordres de grandeur par rapport à WavMark, dont l'algorithme de décodage de la synchronisation est prohibitif sur des contenus non tatoués (le cas dominant dans la pratique). En multi-bit, le modèle attribue presque parfaitement un audio à un modèle parmi 1000, même après éditions.

Enfin, le chapitre analyse en détail les questions de *sécurité* et d'*intégrité* liées à l'ouverture du code source : que peut faire un attaquant qui accède au générateur ? au détecteur ? aux deux ? Cette analyse, qui dépasse le cadre purement expérimental, met en évidence un défi structurel pour le tatouage en source ouverte : la distribution du détecteur permet à des utilisateurs malveillants de concevoir des attaques adaptées, ce

qui rappelle la tension classique entre transparence et robustesse en cryptographie.

Tatouage latent des modèles de langage audio

La quatrième contribution, publiée à ICASSP 2025, étend le tatouage aux modèles génératifs eux-mêmes, et non plus seulement à leurs sorties.

Le problème est le suivant. Le tatouage post-hoc, tel qu'AudioSeal, fonctionne très bien lorsqu'on contrôle le pipeline d'inférence (par exemple, derrière une API). Mais lorsqu'un modèle est *open-source*, un utilisateur peut simplement commenter la ligne de code qui applique le tatouage, et générer des contenus non marqués. Dans le domaine de l'image, des solutions ont été proposées qui ré-entraînent le décodeur pour qu'il produise directement des images tatouées (Stable Signature). Mais dans le domaine audio, l'écosystème est différent : il est commun et peu coûteux d'entraîner ou de remplacer le décodeur (vocodeur), ce qui rend la stratégie "tatouage dans le décodeur" insuffisante : un attaquant peut substituer un décodeur non tatoué.

La proposition est de remonter *en amont* du décodeur : tatouer le modèle de langage audio lui-même, c'est-à-dire la partie qui génère les tokens latents. L'intuition est la suivante : si l'on dispose d'un tatouage qui survit à la tokenisation par le codec neuronal, alors un modèle entraîné sur des données tatouées apprendra naturellement à générer des tokens dont la décodification produit un signal tatoué, indépendamment du décodeur ou du conditionnement utilisé.

La méthode procède en trois étapes. D'abord, on entraîne un générateur et un détecteur AudioSeal modifiés pour être *robustes à EnCodec* : pendant l'entraînement du détecteur, l'audio tatoué est passé à travers EnCodec (encodage puis décodage) une fois sur deux, de sorte que le détecteur apprend à reconnaître la marque même après tokenisation. Ensuite, on tatoue l'intégralité des données d'entraînement du modèle de langage avec ce générateur. Enfin, on entraîne MusicGen, sur ces données tatouées, par la procédure standard de prédiction du token suivant.

Le résultat est un modèle de langage audio dont les sorties sont, par construction, tatouées. Les évaluations montrent trois propriétés remarquables. Premièrement, la *qualité* de la génération n'est pas dégradée : sur les métriques FAD, KL, CLAP-score, le modèle entraîné sur données tatouées est statistiquement indistinguable d'un modèle entraîné sur données propres. Deuxièmement, la *détection* fonctionne avec une confiance élevée sur

les sorties du modèle, pour différents conditionnements (texte, audio de référence) et différents algorithmes de décodage. Troisièmement, et c'est le point crucial pour le scénario open-source, le tatouage *résiste aux modifications du modèle* : changement du décodeur (par exemple remplacement par un autre vocodeur, ou usage de MBD), fine-tuning du modèle de langage sur de nouvelles données, tout cela ne supprime pas la marque. Le tatouage est donc inscrit *dans les poids* du modèle, et non dans une post-traitement détachable.

Cette contribution démontre qu'un modèle peut être à la fois *open-source* et *traçable*, ce qui était considéré comme une tension difficilement résoluble dans le domaine audio.

Synthèse et perspectives

Pris ensemble, les quatre articles de cette thèse constituent une exploration cohérente du pipeline génératif audio fondé sur des représentations discrètes, en abordant à la fois la *qualité* (Multi-Band Diffusion), la *contrôlabilité* (MusicGen-Stem), et la *traçabilité* (AudioSeal puis tatouage latent).

Sur le plan de la modélisation, les contributions montrent qu'il est possible d'aller au-delà du paradigme "un texte, une forme d'onde" en raffinant chaque étage du pipeline : un décodage de meilleure qualité par diffusion, et un schéma de tokenisation et de conditionnement plus structuré permettant la génération et l'édition multi-pistes.

Sur le plan de la sécurité, les contributions démontrent que le tatouage peut être intégré comme un composant à part entière du pipeline génératif, et non comme une rustine post-hoc fragile : AudioSeal offre une détection localisée temps réel, et l'approche par tatouage latent étend cette propriété aux modèles eux-mêmes, ouvrant la voie à des modèles open-source responsables.

Plusieurs questions ouvertes structurent les perspectives. La première concerne les codecs neuronaux eux-mêmes : la compression actuelle approxime la forme d'onde de manière purement géométrique, si bien que des signaux perceptivement équivalents (par exemple x et $-x$) occupent des régions distantes de l'espace latent. Concevoir des espaces latents invariants aux équivalences perceptives, sans sacrifier le taux de compression, reste un problème ouvert dont la résolution simplifierait considérablement l'entraînement des modèles de langage associés.

La seconde concerne la perte de fidélité induite par la quantification vectorielle résiduelle (RVQ). Augmenter mécaniquement le nombre de *codebooks* ne résout pas le problème, car les derniers tendent à ne capturer que du bruit résiduel et deviennent exponentiellement plus difficiles à modéliser. Des approches qui modélisent explicitement les dépendances entre niveaux de quantification (Qinco), ou qui remplacent la quantification par un objectif de diffusion dans l'espace latent, pourraient combiner les avantages des modèles autorégressifs et de diffusion.

La troisième concerne la contrôlabilité : pour atteindre les standards des professionnels de la musique, il est nécessaire d'aller au-delà du texte comme modalité de contrôle. L'intégration de représentations musicales structurées (annotations symboliques, pistes séparées, légendes alignées temporellement) dans les pipelines d'entraînement et d'inférence est une direction prometteuse, qui nécessitera des jeux de données couplant audio et métadonnées musicales fines.

Sur le versant sécurité, la question de la *distribution sécurisée* des générateurs et détecteurs de tatouage reste un défi. Diffuser le détecteur publiquement permet la vérification indépendante, mais offre aussi aux adversaires les moyens de concevoir des attaques. Le verrouiller préserve la robustesse, mais sape la crédibilité du système comme mécanisme de responsabilité. Des approches issues de la cryptographie (preuves à divulgation nulle, engagements cryptographiques, validation fédérée) pourraient offrir un compromis entre vérifiabilité publique et résistance aux attaques.

Enfin, le tatouage pour l'*attribution* ouvre un horizon plus large : relier un contenu généré aux données d'entraînement qui y ont contribué pourrait fonder des mécanismes de rémunération équitable des créateurs humains, à la manière des royalties dans l'industrie musicale. Cette piste, à l'intersection de l'ingénierie audio, du droit et de l'éthique numérique, dessine un horizon où la créativité humaine et la créativité algorithmique peuvent coexister et se reconnaître mutuellement.

En somme, cette thèse défend une thèse simple : la génération audio responsable n'est pas un compromis entre performance et sécurité, mais un effort conjoint sur l'ensemble du pipeline. Les contributions présentées posent les fondations techniques pour des modèles à la fois plus capables, plus contrôlables, et plus traçables.

General Introduction

The rapid progress of deep learning over the last decade has transformed the field of audio processing. Before this shift, most audio analysis and synthesis relied on carefully engineered signal processing pipelines such as Fourier or wavelet transforms for feature extraction, Gaussian mixture models and hidden Markov models for speech recognition, concatenative synthesis for text-to-speech, and rule-based or symbolic approaches for music generation. While these methods were well understood and computationally efficient, they depended heavily on hand-crafted features and task-specific heuristics, which often limited flexibility, generalization, and perceptual quality.

The introduction of deep neural networks, first in supervised speech recognition and later in generative tasks, changed this landscape. Models began to learn directly from raw or minimally processed audio, discovering representations optimized for the task at hand rather than relying solely on human-designed features. Breakthroughs in representation learning, sequence modeling, and neural audio codecs enabled machines to capture long-range temporal dependencies and subtle acoustic structures with unprecedented accuracy. This has led to speech synthesis systems capable of producing human-like voices, music generation tools that compose entire songs from a short prompt, and general-purpose audio models able to operate across multiple modalities.

The same capabilities that make generative audio models powerful also raise important challenges. As their quality improves, generated signals become increasingly indistinguishable from real recordings, creating risks in areas such as misinformation, impersonation, and copyright infringement. This duality of unlocking new creative possibilities while

introducing potential harms has shifted part of the research community's focus from asking whether we can generate high-quality audio to exploring how we can do so safely and responsibly. Among the proposed solutions, digital watermarking has emerged as a promising approach to reliably identify or trace AI-generated audio, even when altered by user modifications or platform processing.

Within this evolving landscape, this thesis focuses on two complementary research axes that have gained importance. The first is the development of generative audio models that combine high quality, controllability, and efficiency. Among the various approaches in the literature, particular attention is given to audio language models, with the aim of improving the quality of their outputs and enhancing their controllability.

The second axis concerns the responsible deployment of such models. If generative systems are to be adopted widely in creative industries, communication, and accessibility tools, there must be mechanisms to authenticate and trace the origin of generated content. Digital watermarking addresses this need by embedding information into the generated audio that can be detected later, even if the signal has undergone common transformations such as compression, equalisation, or resampling. The challenge is to design watermarking methods that are both robust and imperceptible, while remaining compatible with the architectures of audio language models and meeting the constraints of open-sourcing.

The objective of this thesis is to contribute to both of these axes, exploring how to improve audio generative models based on discrete latent representations, and how watermarking techniques can be integrated into their pipelines to support safer audio generation. The work presented here is guided by the following questions:

— **How can we improve the quality and controllability of audio language models based on discrete latent representations?**
— **How can we design watermarking methods that are robust, imperceptible, localized and compatible with open-sourced audio language models?**

Contributions

The main contributions of this thesis are based on the following 4 published articles:

Main Contributions

- **"From Discrete Tokens to High-Fidelity Audio Using Multi-Band Diffusion"** published at Neurips 2023 presents a method to improve the quality of generative audio models based on discrete latent representations, by introducing a multi-stage diffusion process and a novel data processing technique.
- **"MusicGen-Stem: Multi-stem music generation and edition through autoregressive modeling"** published at ICASSP 2025 proposes a method that improves the controllability of audio language models by introducing a multi-stem music generation and edition method based on autoregressive modeling.
- **"Proactive Detection of Voice Cloning with Localized Watermarking"** published at ICML 2024 introduces an efficient audio watermarking method that preserves the audio quality while being both robust and localized.
- **"Latent Watermarking of Audio Language Models"** published at ICASSP 2025 builds on the previous paper to develop a watermarking method to watermark audio language models in an open-source compatible way and study the trade-offs between robustness and open-sourceability.

Outline

This thesis is structured around two main axes in the field of audio processing and generation. It begins with a General Bibliography that lays the foundation by introducing essential concepts in audio processing, machine learning, audio generative models, and evaluation metrics. The first part of the thesis focuses on improving generative audio models through discrete latent representations. Chapter 5 introduces Audio Language Models and provides a comprehensive review of the current state-of-the-art methods. Building on this, Chapter 6 proposes an enhanced approach for decoding discrete audio tokens using raw-waveform diffusion models. Chapter 7 then presents MusicGen-Stem, a method designed to increase the controllability of audio language models, enabling multi-stem music generation and editing with greater flexibility.

The second part of the thesis shifts focus to the development of audio watermarking techniques. Chapter 8 explores the motivations and challenges associated with watermarking, accompanied by a review of existing methods, particularly in the context of audio. Chapter 9 introduces AudioSeal, a state-of-the-art watermarking approach that achieves robustness, imperceptibility, and precise localized detection of watermarked content. Finally, Chapter 10 extends this work by presenting a method to watermark audio language models in a manner compatible with open-source frameworks, building on the principles established by AudioSeal.

General Bibliography

Chapter 1

Digital audio processing

Audio processing is a subfield of digital signal processing restricted to audio signals. An audio signal $x \in \mathbb{R}^T$ is a temporal series called waveform, physically it represents the air pressure variations compared to the ambient pressure. A null signal represents constant pressure which is effectively silence. The sample rate f_r is the number of points that represent 1 second. This value is critical to the audio representation because it relates to the maximal frequency that can be present in a signal. The Shannon-Nyquist criterion (Shannon, 1948) sets the maximum frequency that can be present in an audio signal with sample rate f_r at $f_r/2$. Most humans are completely incapable of hearing frequencies above 20kHz, this is why sample rates are rarely over 48 kHz. For example, compact disc (CD) sample rate is 44.1kHz. Although, in digital signal processing, the waveform is a discrete-time sampled signal, we will, throughout this manuscript, refer to it as a continuous representation by a slight abuse of terminology.

1.1 Fourier transform

1.1.1 Definition

The Fourier transform is a powerful tool for analyzing audio signals, allowing us to decompose a signal into its constituent frequencies. For a continuous-time audio signal

$x(t)$, the Fourier transform is defined as:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt$$

where $X(f)$ is the frequency-domain representation of the signal, and f is the frequency. In the context of audio signals, the Fourier transform can be used to analyze the spectral content of a signal, revealing the distribution of energy across different frequencies. This information can be useful for a variety of applications, including audio filtering, compression, and analysis. For discrete-time audio signals, the discrete Fourier transform (DFT) is often used instead:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$$

where $X[k]$ is the DFT of the signal, and N is the number of samples. The Fast Fourier Transform (FFT) is an efficient algorithm for computing the DFT, and is widely used in many audio processing applications.

1.1.2 Properties of the Fourier transform

The Fourier transform has several important properties that make it useful for analyzing audio signals:

— Linearity: The Fourier transform is linear, meaning that the transform of a sum of signals is equal to the sum of their individual transforms.

— Time-shifting: A time shift in the input signal results in a corresponding phase shift in its Fourier transform.

These properties make the Fourier transform a powerful tool for analyzing and manipulating audio signals.

1.2 Short-time Fourier transform (STFT)

While the Fourier transform provides valuable information about the spectral content of an audio signal, it has some limitations. In particular, it assumes that the signal is stationary, meaning that its spectral characteristics do not change over time. However, many audio signals are non-stationary, with spectral characteristics that evolve over time.

To address this limitation, the short-time Fourier Transform (STFT) (Schnupp et al.,

2011) was developed. The STFT is a variant of the Fourier Transform that analyzes a signal in short overlapping segments, allowing us to capture the evolution of spectral characteristics over time.

1.2.1 Definition of the STFT

The STFT of a continuous-time audio signal $x(t)$ is defined as:

$$X(\tau, f) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-j2\pi ft}dt$$

where $w(t)$ is a window function that selects a short segment of the signal centered at time τ, and f is the frequency.

In practice, the STFT is often computed using a discrete-time signal $x[n]$ and a discrete-time window function $w[n]$. The discrete-time STFT is given by:

$$X[m, k] = \sum_{n=0}^{N-1} x[n]w[n - mH]e^{-j2\pi kn/N}$$

where m is the frame index, k is the frequency bin index, N is the number of samples per frame, and H is the hop size between frames.

1.2.2 Properties of the STFT

The STFT has several important properties that make it useful for analyzing audio signals (Spanias et al., 2007):

— Time-frequency resolution: The STFT provides a trade-off between time resolution and frequency resolution, depending on the choice of window function and hop size.
— Non-stationarity: The STFT can capture the evolution of spectral characteristics over time, making it suitable for analyzing non-stationary signals.
— Linearity: The STFT is linear, meaning that the transform of a sum of signals is equal to the sum of their individual transforms.

These properties make the STFT a powerful tool for analyzing and manipulating

audio signals, particularly in applications such as speech processing, music analysis, and audio effects processing.

1.2.3 Spectrogram Representation

A common way to visualize the STFT is through a spectrogram representation, which plots the magnitude of the STFT coefficients as a function of time and frequency. This provides a visual representation of the signal's spectral characteristics over time, allowing us to identify patterns and trends in the data.

For example, a spectrogram of a speech signal might show the evolution of formant frequencies over time, while a spectrogram of a music signal might show the distribution of energy across different frequency bands.

1.2.4 Mel-frequency cepstral coefficients (MFCCs)

Mel-frequency cepstral coefficients (MFCCs) (Rabiner and Schafer, 1978) are a widely used feature extraction technique in speech processing and music information retrieval. They are designed to capture the spectral characteristics of an audio signal in a way that aligns with human auditory perception, making them particularly effective for tasks such as speech recognition and music classification.

The computation of MFCCs begins by dividing the audio signal into short overlapping frames (typically 20-30 milliseconds). Each frame is then transformed into the frequency domain using the Fast Fourier Transform (FFT). The resulting spectrum is filtered using a set of triangular filters spaced according to the mel scale, a perceptual scale that reflects the human ear's sensitivity to different frequencies. The mel-scale filter outputs are then subjected to logarithmic compression to mimic the human ear's logarithmic perception of loudness. Finally, a Discrete Cosine Transform (DCT) is applied to decorrelate the filter bank energies and produce a compact set of coefficients. This transformation also helps separate the source (excitation) and filter (vocal tract) characteristics of speech, allowing the resulting coefficients to primarily represent the vocal tract shape that conveys phonetic information.

MFCCs are valued for their robustness to noise and their ability to provide a compact representation of audio signals. Because they are designed to mimic human auditory

perception, they are particularly relevant for applications such as speech recognition and music analysis, where understanding and interpreting audio content is the primary goal.

1.3 Audio Compression

The basic numerical waveform representation of audio signal that represents every point independently is very costly in storage. Audio compression algorithms (called codecs) try to reduce this consumption of storage. There are two families of such algorithms. One which consists in algorithms that can reduce the size of the audio files while being capable of retrieving the original signals intact (lossless codecs). On the other hand, lossy codecs try to remove information that is less noticeable to achieve higher compression rate.

Lossless codecs

Lossless codecs such as FLAC (Xiph.Org Foundation, 2001) or ALAC typically use linear predictive coding (Makhoul, 1975) which consists in using a simple linear function fitted on the audio. The difference between the prediction and the signal is encoded using Rice Coding (Sayood, 1996). This family of codecs typically achieves compression gains of around 50%.

Lossy codecs

Usual lossy audio codecs such as MP3 or AAC rely on decomposing the signal into a sum of sinusoidal functions using the discrete Fourier transform. Then a psychoacoustic model will remove some of the sinusoids to keep only the most important components. For example, it removes components with very small amplitude or erases components that are too close to another and would not be distinguished by the human ear. This approach, while resulting in a quality deterioration, can achieve compression rate of 30x when starting from high resolution audio such as 24 bits 44.1kHz.

Chapter 2

Machine learning basics

In this section we introduce the basic concepts of machine Learning that are necessary to understand the rest of the manuscript.

2.1 Definitions

Machine learning (ML) is a subfield of Artificial Intelligence (AI) that focuses on the development of algorithms and statistical models that enable computers to perform tasks without explicit instructions. Instead, ML systems learn from data, identifying patterns and making decisions based on the information they have been trained on.

Instead of defining a procedure to solve a task, ML algorithms are trained on a dataset, which is a collection of examples that represent the problem to be solved. The objective is to build a model that can generalize from the training data to make predictions or decisions on new, unseen data. The model is typically represented as a parametric mathematical function that maps input data to output predictions. Parameters of such a model are called weights.

To train such a model, one has to define a cost function that measures the error of the model on the training data. This function is called the loss function. It should be differentiable to allow for the use of gradient descent optimization algorithms such as

stochastic gradient descent SGD or Adam (Kingma and Ba, 2015).

2.2 Supervised learning

Supervised learning is a type of machine learning where the model is trained on a labeled dataset, meaning that each example in the training set is paired with a corresponding label or target value. The goal of supervised learning is to learn a mapping from inputs to outputs, allowing the model to make predictions on new, unseen data. Supervised learning can be used for both regression and classification tasks. In regression, the model predicts a continuous output value, while in classification, the model predicts a discrete class label. Typically, a loss function for regression problems is a function that will measure the distance between the predicted value and the true value, such as mean squared error (MSE).

$$\mathcal{L}_{\text{MSE}}(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.1}$$

where y is the true value, $\hat{y}$ is the predicted value, and N is the number of samples.

For classification tasks (Krizhevsky et al., 2012; LeCun et al., 1989), the loss function is often a cross-entropy loss that measures the difference between the predicted probability distribution and the true distribution. The ground truth is often represented as a one-hot encoded vector, where the true class is represented by a 1 and all other classes are represented by 0.

$$\mathcal{L}_{\text{cross-entropy}}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{2.2}$$

where C is the number of classes, y is the true distribution, and $\hat{y}$ is the predicted distribution.

While supervised learning is a powerful approach, it requires a large amount of labeled data specific to a task, which can be expensive and time-consuming to obtain.

2.3 Unsupervised learning

Unsupervised learning is a type of machine learning where the model is trained on an unlabeled dataset, meaning that the examples in the training set do not have corresponding labels or target values. The goal of unsupervised learning is to discover patterns or

structures in the data without any prior knowledge of the labels. Unsupervised learning can be used for various tasks, such as clustering, dimensionality reduction, and anomaly detection.

Recently, the term self-supervised learning has been used to describe a specific type of unsupervised learning where the model is trained on a pretext task that generates pseudo-labels from the data itself. This approach allows the model to learn useful representations of the data without requiring explicit labels. It has been successfully applied in various domains, including computer vision (Chen et al., 2020; Oquab et al., 2024) and natural language processing (Brown et al., 2020a; Devlin et al., 2019b). For tasks ranging from image generation (Ho et al., 2020) to video understanding (Bardes et al., 2024), self-supervised learning has been shown to be effective in learning rich representations that can be fine tuned for specific downstream tasks.

Chapter 3

Audio Representation Learning

This chapter will go over the current literature in Audio representation learning. As is the case for most topics in Audio processing, representation learning of audio signal is a field that has been driven mainly by the speech community. This chapter describes first general audio representation learning and then will focus more specifically on discrete representations of audio signals as it is a key aspect of Audio Language Models.

3.1 Supervised learning

Supervised Learning consists of using labeled data to train machine learning models. This approach has proven in multiple domains to be able to extract meaningful representations of data in various domains. In the case of audio, the standard task for supervised learning has been Automatic Speech Recognition (ASR). This task has long been the primary focus of the community. However, this task seems to have reached a plateau in terms of ASR performances (word error rate, etc.). Recent models trained for ASR on very large labeled data sets such as Whisper (Radford et al., 2023) have proven to be able to perform very well on a variety of speech analysis tasks unrelated to ASR.

While training models on large scale datasets on supervised tasks has proven to be a very efficient way to learn rich features for audio supervised learning still poses two major limitations. First, it requires a huge amount of labeled data which is not always available.

Second, while in the case of speech analysis the task of transcribing audio to text is a very sensible and meaningful objective it is not easy to find a similarly fine and extensively annotated task in the case for instance of music or general audio.

3.2 Self Supervised Learning

Dealing with the lack of annotated data is a classic problem in machine learning called unsupervised learning. Recently most of the literature on unsupervised learning is often called self supervised with the idea that one can extract labels from the data itself and then use those extracted labels for a supervised task.

3.2.1 General Audio Representations

In parallel of the progress in Speech representation models, there was also some progress in the general audio representation domains with a multitude of methods. One important innovation was inspired by the work Clip (Radford et al., 2021) that learns a joint embedding for both images and text. There was recently an increase in interest to learn this type of multimodal embeddings with text/audio models such as Mulan (Huang et al., 2022a) or CLAP (Elizalde et al., 2023), another approach, Image Bind (Girdhar et al., 2023) used a frozen image space as an anchor to bind embeddings in a multitude of modalities including audio, text, depth-images..

Beyond these cross-modal methods, architectures focused purely on general audio features also emerged. The Audio Spectrogram Transformer (AST) (Gong et al., 2021) introduced a pivotal shift by becoming the first convolution-free, purely attention-based model for audio classification, processing the 2D audio spectrogram as a sequence of patches, similar to how Vision Transformers handle images. Similarly impactful, the openL3 framework (derived from "Look, Listen and Learn") (Arandjelović and Zisserman, 2017a;b) utilized a self-supervised approach, training coupled audio and visual networks by solving an Audio-Visual Correspondence (AVC) task on large amounts of unlabeled video to learn robust, general-purpose audio features. More recently, BEATS (Bidirectional Encoder representation from Audio Transformers) (Chen et al., 2023) proposed an iterative self-supervised pre-training framework that uses a discrete label prediction objective, relying on a learned acoustic tokenizer to encourage the model to abstract high-level audio semantics and discard redundant details.

While those approaches allow linking audio to other modalities such as images or text. They suffer from losing the temporal granularity of the audio by generating a unique embedding per audio sample. Those models often rely on very general text descriptions such as "a person speaking" which makes it completely unable to represent the local details within the audio that make it possible to recognize the content of the speech and perform ASR. Other self supervised methods, described in the following section, are designed for ASR and are more focused on local information.

3.2.2 Speech representation learning

When learning to extract representations specific for speech, the goal is to learn representations that capture the phonetic, prosodic, and speaker characteristics of speech signals. Contrary to general audio representation learning, speech representation learning is often more focused on the local details of the audio signal, such as phonemes and words, rather than global characteristics. In Contrastive Predictive Coding (CPC) (van den Oord et al., 2019), the authors build an encoder that extracts a sequence of embedding vectors from the audio signal, then a contrastive loss is applied to distinguish between positive samples (future frames) and negative samples (randomly selected frames). To force models to focus on speech content, the negative samples are selected from the same speaker. This way the model can't rely on speaker identity to distinguish between positive and negative samples.

Recently, speech representation learning has been dominated by discrete representation methods that will be described in Section 3.3.

3.3 Discrete Audio Representations

Discrete representation learning consists in converting audio signal into a sequence of discrete vectors called audio tokens. Those vectors are similar to features of audio representation models but restricted to a finite number of possible features. This discretization while resulting in an information loss has multiple benefits: First as standard with quantization it reduces the memory consumption of storing the representations and is used at scale for similarity search (Douze et al., 2025). Second it allows for using most common sequence modeling algorithms developed for the text modality. Lastly, quantization helps with the disentanglement of features.

The literature on discrete audio representation learning distinguishes between two types of representations called *acoustic tokens* and *semantic tokens*. Acoustic tokens aim at representing the whole audio signal, they aim at being used for reconstruction and are the representations of Neural Audio Codecs (NAC). NAC are extensively described later (see Section 5.2). On the other hand semantic tokens try to incorporate higher level information such as phonetic content for speech. Semantic tokens make the task easier for downstream speech models Lakhotia et al. (2021); Nguyen et al. (2024).

3.3.1 Semantic Tokens

There is a wide variety of techniques to extract discrete semantic representation from audio. These most of the time rely on first an encoder that will extract continuous representation and then a clustering technique that will convert this continuous representation into a fixed number of centroids. Clustering methods can include K-Means, Gumbel softmax quantization layer to sample from centroids in a fully differentiable way or with a vector quantization (VQ) layer.

Supervised Method

As in representation learning in general, the natural strategy when labeled data is available is to use supervised learning. In the case of audio, the most common way to learn semantic representations from labeled data is to use ASR objective. This can simply be done by performing an ad hoc K-Means clustering on the intermediate representation. Another solution proposed in $\mathcal{S}^3$ Tokenizer (Du et al., 2024) is to directly train an ASR model with a VQ layer inside the model.

Contrastive methods

Contrastive methods for learning discrete semantic representations from audio involve training a model to differentiate between similar and dissimilar audio samples. This approach has been successfully applied in models such as wav2vec 2.0 (Baevski et al., 2020b) and VQ-wav2vec (Baevski et al., 2020a).

In wav2vec 2.0, after an encoder layer, features are quantized using a product quantization layer (Jegou et al., 2010) then, after masking some features, the sequence is fed to a transformer. After this stage, a contrastive loss is applied to identify representations that

have been masked from negative ones taken from the same utterance. A diversity loss is added to maximize codebook utilization. While wav2vec 2.0 uses quantization layer, it is mostly used to design the training objective.

VQ-wav2vec is very similar to wav2vec 2.0, is not using discrete representation as a training objective but directly aims at creating high quality discrete representations for downstream language modeling tasks. Key differences are the quantization method and the fact that the model will try to distinguish future steps from other examples instead of randomly masked steps.

Predictive methods

Predictive methods are usually based on Bert (Devlin et al., 2019a) style language modeling. Models such as HuBERT (Hsu et al., 2021) or WavLM (Chen et al., 2021b) use this approach. They build an encoder that will extract a sequence of embedding vectors then, during training, after masking a proportion of the vectors, the model will try to predict pseudo labels from the audio. The training has a few stages, pseudo labels are obtained from K-Means clustering over the MFCC in the first stage and over the intermediate representation in later stages. This process of predicting pseudo labels from a masked sequence forces the model to understand the high level semantic of the sequence.

3.3.2 Disentanglement

The challenge of representation learning is to transform raw data into interpretable and structured representations. One key aspect of that is to separate features that might not be linked. This is referred to as disentanglement. In the case of speech it seems natural to separate various factors that impact audio recording such as content (which words are said), prosody (the way it's said), speaker identity, acoustic of the room.. Ideally a good representation should include all those parameters but in a non-intricate way. Discrete representation has been used as a tool to achieve features disentanglement.

Implicit disentanglement

The information compression caused by the quantization step forces the model to keep only useful information for the training task. This helps when needing to separate different attributes of audio. This phenomenon has been studied in the case of speech

resynthesis (Polyak et al., 2021a), where authors use different phoneme level discrete representation of speech (Hubert (Hsu et al., 2021), CPC (van den Oord et al., 2019), VQVAE (Garbacea et al., 2019)) combined with a quantized f_0 and speaker embedding. Experiments show that the speech representations are disentangled from f_0 and speaker identity while none of the representations have been trained explicitly. This allowed authors to perform 0-shot voice conversion.

Explicit disentanglement

Contrary to implicit disentanglement where compression alone results in removing some aspects of the representation, some model explicitly aim at disentangling different features. One strategy used to remove elements from the representations is to use a gradient reversal layer, the principle is to train a model that takes as input the representation and tries to predict the feature that one wants to remove from the representation. This creates an adversarial setup where the encoder is optimized to produce representations from which the unwanted feature (e.g., speaker identity) cannot be accurately predicted, effectively forcing it to discard that information while retaining other relevant features. This is exemplified with models such as FACodec from Natural speech 3 (Ju et al., 2024) or SSVC (Martín-Cortinas et al., 2024).

Other methods to leverage the information bottleneck to separate different features include perturbing the features in the input audio (Guo et al., 2024) or using explicit disentangled objective such as source separation (Bie et al., 2025).

The use of disentangled representations can be particularly useful to improve controllability in audio generative models, a class of models that will be introduced in the next chapter.

Chapter 4

Audio generative Models

Generative modeling is the task aiming at sampling from a data distribution $\mathcal{D}$. In this section we will go over families of generative models that most commonly are used for audio generation. We will make a distinction between two types of those generative models. The first type is what is referred to as vocoders, those models takes as input a low level audio representation (e.g. mel-spectrogramme, MFCC, SSL representation..) and generate a corresponding audio signal. The second type are the creative models, those are models that can generate audio either without conditioning or conditioned with high level semantic representation (e.g. text, speaker embeddings etc..).

4.1 Vocoders

Vocoders were traditionally trained with the simple objective of constructing a waveform using a Neural Network conditioned on an input c. Those models were trained using simple objectives such as the L1 or L2 distance in the waveform domain or some distances in the STFT domain. These simplistic approaches resulted in very poor audio quality and were replaced by more advanced methods. In recent years two main strategies have been adopted for modern vocoders. The first relies on Generative Adversarial Networks (GAN) and the second relies on diffusion models.

4.1.1 Gan Based Vocoders

General framework

For most generative tasks it is hard to define an objective function that can evaluate if a sample generated by the model G fits in the data distribution $\mathcal{D}$ as this problem is as hard as the sampling problem. The idea of GAN is to jointly train a second model D called the discriminator that will learn this objective.

The model D is simply trained to solve a classification where it has to decide whether a datapoint belongs to the data distribution or is generated by the model. It is trained with simple binary cross entropy loss.

The model G is then trained to generate outputs that the discriminator classifies as belonging to the data distribution. This is done by computing the cross entropy loss between the discriminator's output and the desired class(Goodfellow et al., 2014). This results in the following loss:

$$L_G = -\log(D(G(z))) \tag{4.1}$$

The two models are then trained in an alternate way. While GAN allows to deal with the lack of clear definition for an objective function they usually suffer from two limitations. First they are very unstable to train and the equilibrium between the performances of the two models is very important such that important information can flow from the gradients of the discriminator to the generator. Second they are prone to mode collapse. This issue arrives when the generator gets good at sampling only a very small subspace of the data distribution making it a poor way to sample from the complete distribution.

GAN Vocoders

GAN based vocoders combine standard reconstruction losses with the GAN objective.

Different approaches, often conditioned on mel-spectrograms, have been explored for this task, including GAN-based methods such as HiFi-GAN (Kong et al., 2020; Kumar et al., 2019). Polyak et al. (2021b) used HiFi-GAN on other types of conditioning. This method generates speech conditioned on low bit rate representations learned from self-supervised methods such as HuBERT (Hsu et al., 2021) VQ-VAE (van Niekerk et al.,

2020) or CPC (van den Oord et al., 2019) together with the fundamental frequency f_0 and a speaker embedding. By using only a few centroids for clustering, the speech content representation becomes largely disentangled from the speaker and the fundamental frequency (f_0), enabling controllable speech generation.

Diffusion Vocoders

Diffusion-based vocoders are inspired by the recent success of diffusion for image generation (Dhariwal and Nichol, 2021; Ho et al., 2020; Ramesh et al., 2022; Saharia et al., 2022). Kong et al. (2021) introduced Diffwave, a diffusion-based vocoder, that applies the vanilla diffusion equations to waveform audio. Compared with the adversarial approach, diffusion offers a simpler L2 loss objective, and stable training. PriorGrad (Lee et al., 2022) is an extension of Diffwave that uses non standard Gaussian noise in the diffusion process. The authors extract the energy of the conditioning mel-spectrogram and use it to adapt the prior noise distribution to the target speech. Wavegrad (Chen et al., 2021a) is similar but uses conditioning on continuous noise levels instead of discrete ones. This allows the model to perform the sampling using any noise schedule with a single training. Takahashi et al. (2023) look at singing voices, which is a more complex distribution than standard read speech due to wider spectrum, and increased diversity. Inspired by super-resolution cascaded techniques from image diffusion (Ho et al., 2022), they used hierarchical models. The first diffusion model synthesises at a low sampling rate while later ones, conditioned on the output of their predecessor, perform upsampling. This process can yield high-quality, high-resolution audio samples. Recent work (Pascual et al., 2022) applies diffusion to generating full band audio at high sampling rate, although the proposed method allows for unconditional generation, and flexible style transfer, it remains limited to a narrow range of audio modalities.

4.2 Creative models

The previous section went over **vocoders**, *i.e.* models that can take low level audio representations such as mel spectrogram or acoustic features to generate waveform audio. **Creative models**, *i.e.* models that either unconditionally or with very high level concepts conditioning will generate either audio or lower level representations. This section briefly goes over the main families of creative models that are used in the audio domain excluding Audio Language Models (ALM) that will be described in detail in Chapter 5.

4.2.1 Text to Speech

Text to Speech (TTS) is the task of generating speech from text. This task has been the focus of the speech community for decades and has seen a lot of progress. TTS has long been dominated by concatenative methods, which rely on a database of pre-recorded speech segments. The community then moved to more statistical methods, such as Hidden Markov Models (Tokuda et al., 2002) and Gaussian Mixture Models (Zen et al., 2013) which model the relationship between text and speech parameters. Later, those model were replaced by neural networks (Zen and Senior, 2014) and recently by end-to-end neural networks such as Tacotron (Shen et al., 2018; Wang et al., 2017) or FastSpeech (Ren et al., 2019; 2021). Most recent TTS models manage to generate high quality speech with a very natural prosody and intonation. Models such as Kokoro [1] manages to do so while using less than 100M parameters.

4.2.2 Music Generation

Music generation is the task of generating music, either conditioned on a text description or unconditionally. Symbolic music generation, *i.e.* the task of generating music scores, has been a long-standing task in the music community, with early approaches using rule-based systems and Markov models (Boulanger-Lewandowski et al., 2012). Deep Learning methods have also been applied to this task with models such as MusicVAE (Roberts et al., 2018), MuseGAN (Dong et al., 2018) or Music Transformer (Huang et al., 2019).

Alternatively, another part of the community has focused on generating audio music, *i.e.* producing raw audio waveforms or spectrograms instead of scores or MIDI. Pioneering models like WaveNet (Oord et al., 2016) and SampleRNN (Mehri et al., 2017) demonstrated that neural networks could synthesize realistic audio by modeling each sample conditioned on previous ones, while NSynth (Engel et al., 2017) introduced neural audio synthesis capable of interpolating between instruments.

More recent approaches based on Audio Language Models or latent diffusion models will be discussed in Chapter 5.

1. https://github.com/nazdridoy/kokoro-tts

4.3 Evaluation

Evaluation of generative models is a complex task, especially in the audio domain where subjective quality assessment plays a significant role. This section will discuss the various methods used to evaluate generative models, focusing on both objective and subjective metrics with or without reference audio.

4.3.1 Objective Metrics

Objective metrics are quantitative measures used to evaluate the performance of generative models without human intervention. These metrics can provide a quick assessment of model quality, but they may not always align with human perception. Most of the objective metrics are based on the comparison of the generated audio with a reference audio, if available.

Mean Squared Error (MSE)

Mean Squared Error (MSE) is a simple objective metric that measures the average squared difference between the generated audio and a reference audio. MSE is calculated as follows:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{4.2}$$

where x_i is the reference audio sample, y_i is the generated audio sample, and N is the number of samples. MSE is easy to compute and provides a straightforward measure of the difference between two audio signals. However, it does not account for perceptual differences and may not correlate well with human judgment of audio quality.

In some cases, MSE can be replaced by other distance metrics such as the L1 norm (Mean Absolute Error).

Signal-to-Noise Ratio (SNR)

Signal-to-Noise Ratio (SNR) is a widely used objective metric that measures the ratio of the signal power to the noise power in the generated audio. SNR is calculated as follows:

$$\text{SNR} = 10\log_{10}\left(\frac{\text{Signal Power}}{\text{Noise Power}}\right) \tag{4.3}$$

where Signal Power is the power of the generated audio signal and Noise Power is the power of the noise component (i.e. the difference between original and generated signal). SNR is a simple and effective metric for evaluating the quality of generated audio, but it does not capture all aspects of audio quality, such as perceptual differences.

Perceptual Evaluation of Speech Quality (PESQ)

Perceptual Evaluation of Speech Quality (PESQ) is an objective metric specifically designed for evaluating the quality of speech signals. PESQ compares the generated audio with a reference audio and provides a score that correlates with human perception of speech quality. PESQ is calculated using a model that simulates human auditory perception, taking into account factors such as frequency response and temporal masking.

Visqol

ViSQOL (Chinen et al., 2020) is a full-reference objective metric that estimates the Mean Opinion Score by comparing a degraded signal $d(t)$ with a clean reference $r(t)$. Both are first transformed into gammatone spectrograms (Schnupp et al., 2011) $x_r(t, f)$ and $x_d(t, f)$ using Equivalent Rectangular Bandwidth (Glasberg and Moore, 1990) filters to model human auditory frequency resolution.

The spectrograms are segmented into time–frequency patches P_r, P_d, which are aligned via cross-correlation to account for local timing differences.

For each aligned patch, the Neurogram Similarity Index Measure is computed:

$$\text{NSIM} = \frac{2\mu_r\mu_d + C_1}{\mu_r^2 + \mu_d^2 + C_1} \cdot \frac{2\sigma_{rd} + C_2}{\sigma_r^2 + \sigma_d^2 + C_2},$$

where μ_r, μ_d are patch means, σ_r, σ_d are standard deviations, σ_{rd} is the covariance, and C_1, C_2 are stability constants.

Let $\mathbf{s}$ be the mean NSIM values per frequency band. The final MOS prediction is obtained through a polynomial mapping:

$$\widehat{\text{MOS}} = f_{\text{poly}}(\mathbf{s}).$$

Fréchet Audio distance (FAD)

Fréchet Audio Distance (FAD) (Kilgour et al., 2019) is a metric that measures the similarity between two audio distributions, typically used to compare the distribution of generated audio with that of real audio.

FAD is calculated by first extracting features from the audio signals using a pre-trained ssl representation model (see section 3.2). Then, the features are used to compute the mean and covariance of the distributions for both the generated audio and the reference audio.

$$\text{FAD} = \|\mu_g - \mu_r\|^2 + \text{Tr}(\Sigma_g + \Sigma_r - 2(\Sigma_g\Sigma_r)^{1/2}) \tag{4.4}$$

where μ_g and Σ_g are the mean and covariance of the features extracted from the generated audio, and μ_r and Σ_r are the mean and covariance of the features extracted from the reference audio.

Contrary to previous metrics, FAD does not require per example reference audio, but rather compares the overall distribution of features extracted from the audio signals. It can only measure if the statistics of the set of generated audio are similar to the reference set (training set), but it does not provide a per sample measure of perceptual quality. It is often used to measure diversity of generated audio, as it can capture the spread of the generated samples in the feature space.

4.3.2 Subjective Metrics

Subjective metrics involve human listeners who evaluate the quality of generated audio based on their perception. Subjective evaluation is often considered the gold standard for assessing audio quality, as it takes into account the nuances of human perception that objective metrics may miss. However, subjective evaluation can be time-consuming and resource-intensive, as it requires human listeners to listen to and rate the audio samples. Furthermore, subjective evaluation can be influenced by individual preferences and biases, making it less consistent than objective metrics. They also are not consistent across different listeners, which can lead to variability and noisiness in the results.

Similar to objective metrics, subjective metrics can be divided into two categories: with reference audio and without reference audio.

Mean Opinion Score (MOS)

Mean Opinion Score (MOS) is a widely used subjective metric that involves human listeners rating the quality of generated audio on a scale, typically from 1 to 5. The MOS is calculated as the average of the ratings given by multiple listeners. Participants are asked to listen to the generated audio and provide their opinion on its quality, taking into account factors such as clarity, naturalness, and overall enjoyment. The scale goes from 1 (bad quality) to 5 (excellent quality).

4.3.3 MUSHRA Test

The MUSHRA test (International Telecommunication Union, 2015) is a subjective evaluation method that involves human listeners rating the quality of generated audio samples. Unlike MOS, MUSHRA procedure involves presenting listeners with a set of audio samples, including the generated audio and reference audio, and asking them to rate the quality of each sample on a scale from 0 to 100. The MUSHRA test is designed to assess the quality of audio samples in a more controlled and systematic manner than MOS.

Since it requires reference audio it is most suited for evaluating task such as audio restoration or compression.

4.3.4 Other subjective metrics

Subjective evaluation can also be used to assess specific aspects by asking listeners to rate about specific attributes such as how it sounds like a specific speaker, how it sounds like a specific emotion or how it matches a specific text description. This type of very specific evaluation tend to suffer from high noisiness, as non-expert listeners might not be able to easily judge very specific attributes.

Part I

Generative Audio Models from Discrete Representations

Chapter 5

Introduction to Audio Language Models

This chapter introduces all the necessary concepts to understand how audio language models work. It also goes over some examples of use cases for this type of models in multiple fields that can benefit from audio generative models.

5.1 Motivations

5.1.1 Large Language Models

The field of machine learning has been rapidly evolving, and the area that has grown the most in recent years is natural language processing (NLP). This surge has been driven by the emergence of language models (LM). In the text domain, a language model is a probabilistic model designed to estimate the distribution of words.

One of the earliest types of language models was the k-gram model. These models, given a vocabulary of size N, consist of a table that provides the probability distribution of the next word based on statistics collected from a dataset. Despite their simplicity, k-gram models have proven useful for various applications such as spelling correction (Bassil and Alwani, 2012). However, they have inherent limitations when it comes to modeling long

sequences. Firstly, the size of a typical k-gram model grows exponentially with the context size k, making it impractical to increase k and capture the meaning of long sequences. In consequence, they require an exponentially larger amount of data to construct as n increases. Secondly, k-gram models are unable to generalize, as they can only store statistics of sequences that appear in the training dataset.

Machine learning models have overcome some of the limitations of the k-gram models by leveraging deep learning architectures and optimization methods. These machine learning models learn to predict the probability distribution of words without relying on explicitly stored statistics. While this approach originated on text, it is in practice a general approach that can be applied to any sequential data, such as audio or video. Elements of sequences are discrete points in a finite vocabulary and are called tokens.

The training of those models is typically done in a self-supervised manner, where some tokens in a sequence are masked, and the model is trained to predict the masked tokens based on the unmasked ones. For each masked word the model predicts outputs called logits. Logits indicate the likelihood of each word of the vocabulary in an unscaled way. They can be turned into a probability distribution using the softmax function. The objective function compares the output distribution to the actual distribution (probability of 1 for the ground truth word and 0 for all other words in the vocabulary) using cross-entropy loss (Cover and Thomas, 2006), which is a common loss function for classification tasks. Formally, the loss for a single token t in a sequence of tokens s is defined as:

$$l_t = -\log(\hat{p}_{t,s_t}) \tag{5.1}$$

While other masking strategies have been explored (Lewis et al., 2020), the current standard to train modern LMs relies on next token prediction. This strategy consists in predicting every token in a sequence one by one, conditioning the prediction of the next token on all previous tokens. This left-to-right approach mimics human writing patterns. During inference, these models are used autoregressively, predicting words one by one.

Scaling up these models using the transformer architecture (Vaswani et al., 2017) to a large number of parameters and training them on vast amounts of text data has led to the development of large language models (LLMs). These models have demonstrated the ability to model long and complex sequences, matching or outperforming human

performance in writing tasks, and exhibiting capabilities such as reasoning (DeepSeek-AI et al., 2025; OpenAI et al., 2024), summarization (Brown et al., 2020b; Touvron et al., 2023), and code generation (DeepSeek-AI et al., 2025; Rozière et al., 2024).

5.1.2 Applying Language Model Ideas to Audio

Similar to text, audio exhibits a sequential structure that can be very long, making the efficient learning mechanisms of LLMs highly relevant to the field of audio processing. It is natural to consider applying these ideas from the text literature to build audio models capable of generating long and coherent audio sequences. However, a crucial aspect of LLM training is that they predict logits over a finite vocabulary, which is essential to their success. In contrast, the audio domain is a continuous space, where the samples in a sequence are not discrete tokens but rather continuous values.

The first model to adapt next token prediction ideas to the audio domain was WaveNet (Oord et al., 2016). To achieve this, the authors discretized the audio at the sample level into a fixed number of bins, allowing each sample to take one of 256 values (8 bits). They then employed a similar training technique using the softmax function on the output of the model and cross-entropy loss. The architecture was based on dilated convolutions, enabling longer receptive fields for the neural network. To mitigate the need for excessive model passes, the audio sample rate was limited to 8kHz. Despite this reduced audio quality of 8 bits and 8kHz, generating 1 second of audio still required $8,000$ model calls.

The length of typical audio sequences creates significant challenges when working with raw waveforms. The first challenge is the inference computational cost of generating audio sample by sample. The second challenge is the computational cost of using modern attention-based models like transformers, which have a computational complexity of $O(T^2)$, where T is the sequence length. These limitations, both related to the sequence length, hindered the adoption of these ideas until the emergence of neural audio codecs, which have revitalized interest in this area.

5.2 Neural Audio Codecs

In the case of generative modeling using audio language models, neural audio Codecs aim at transforming audio input into a sequence of discrete values called tokens. This

sequence of tokens can then be converted back to audio using a decoder. This transformation to discrete sequences of tokens is effectively reducing the amount of space the data is taking, making them a new form of lossy compression. This section describes in detail the architecture of neural audio codecs, the losses used to train them and some variants that have been proposed in the literature.

Contrary to traditional lossy codecs, neural audio codec do not rely directly on handcrafted psychoacoustic models but on deep learning based auto encoders with quantization mechanisms. Those models are able to compress audio to extremely low bitrates sometimes under 1kbit per seconds for some models (Liu et al., 2024a). They can also be used to transform audio into low-frame-rate discrete sequences which are useful to train audio language models.

5.2.1 Architecture

Neural audio Codecs are usually based on a very simple convolutional structure. Similar to a U-net model (Ronneberger et al., 2015) the decoder is the symmetric of the encoder where downsampling convolutions are replaced by transposed convolutions. No skip connection link the encoder and decoder such that all the information goes to the bottleneck where the quantization will happen. Some models also include recurrent neural networks such as LSTM before and after the quantization stage. In a codec of frame rate cr designed for audio with sample rate sr: The encoder converts audio sequence $s \in \mathbb{R}^T$ to a sequence of embedding vectors $x \in \mathbb{R}^{D \times T_1}$, where D is the embedding dimension and $T_1 = \frac{cr}{sr} T$ is the number of embedding vectors required in the bottleneck to encode x.

The quantization stage is critical and can be performed using multiple methods. The most common is called Residual Vector Quantization (RVQ) (Vasuki and Vanathi, 2006). This method is a cascade of vector quantization (VQ) layers that will be applied to the embedding vectors. The RVQ module is composed of K VQ layers. A VQ layer is simply a set of reference vectors called *codebook*. This layer can approximate a vector by mapping it to the closest vector in the *codebook*. Each VQ layer has a size 2^n where n is the number of bits used to represent the index of the vectors in the *codebook*. Each layer is approximating the residual left after quantizing with the previous one.

During training, each vector in the *codebook* is updated with an exponentially moving average of input vectors map to it (Défossez et al., 2023). A n-bit VQ layer is a *codebook* in

$\mathbb{R}^{2^n \times D}$. It is called n-bit because n is the number of bits required to represent the index of a vector in the *codebook*. It converts a vector $x_t \in \mathbb{R}^D$ into an integer $id_t \in \{0, 1, \dots, 2^n - 1\}$, the output quantized vector is obtained by looking at the id_t-th vector of the *codebook*. The RVQ module transform is said to have K streams if it is composed of K VQ layers. Finally a RVQ layer transforms the vector x_t into K discrete indices $id_{t,k} \in \{0, 1, \dots, 2^n - 1\}$. The final compressed vector q_t can be retrieved by summing the quantized vector outputs of the different VQ layers.

Naturally, the output space of the encoder is not always easy to quantize, *i.e.* it does not necessarily form clusters of vectors. One common way to promote a clustered space it to add a *commitment loss* (van den Oord et al., 2017). The *commitment loss* is simply an l2 loss between the outputs of the encoder x_t and its compressed version q_t. The gradients of this loss are only used to update the encoder weights and does not impact directly the *codebook*).

$$l_{commit} = ||x_t - q_t||^2 \tag{5.2}$$

This loss is encouraging the encoder to produce outputs that are close to the codebook vectors. In practice, this loss promotes the presence of clusters around the codebook vectors in the output space of the encoder.

To make the gradients flow from the decoder to the encoder during the backward pass, it is standard to use the straight-through estimator (Bengio et al., 2013). The idea behind straight through estimator is to approximate the gradient of the quantized embeddings with the gradient before quantization stage. Formally, this is done using the following formula:

$$\hat{q}_t = x_t + (q_t - x_t).\text{detach}() \tag{5.3}$$

Where .detach() denotes the operation of discarding the gradients. The reader should notice that $\hat{q}_t$ has the same value as q_t. However the gradient are now those of x_t, effectively treating the difference as a noise sometimes called quantization noise.

We will denote in the following $\hat{x}$ the output of this quantized auto encoder with input

x.

5.2.2 Losses

While neural audio codec do not rely on an explicit psychoacoustic model, some kind of model of what information is most important to the reconstruction is hidden in the objective functions that are used to train those models. The combination of losses can vary depending on the model but it generally consist in a combination of the following losses:

— Waveform based losses such as l1 or l2 losses.
— Distances in the spectral space for different spectrogram resolutions (Yamamoto et al., 2020)
— Adversarial and feature-matching losses, similar to previous vocoder works (see section 4.1.1).

The balance between those losses is critical to the performances of the final codec. In the paper of EnCodec (Défossez et al., 2023) authors introduce a way to balance the losses with respect to their contribution to the gradients in the optimisation. Effectively, this makes the loss weights values more correlated to the importance of the loss in the optimisation. While use of losses vary from a codec to an other, the use of adversarial and feature matching losses is present across most models. Mimi Codec (Défossez et al., 2024) demonstrated that only using those losses in the context of a speech codec improves the overall quality in human evaluations.

5.2.3 Variants

SoundStream (Zeghidour et al., 2021) and EnCodec (Défossez et al., 2023) are the original general audio codecs. This type of baseline is often refered to as RVQ-GAN codecs. In this section we will go over some of the variants that have been proposed to improve uppon those or specialize to some modalities *e.g.* speech.

Several works have focus on improving the quantization strategy. Parker et al. (2024b) experiment with scalar quantization (Parker et al., 2024a). The idea of scalar quantization is to project the D dimensional vector x_t into an usually larger space where values will be restricted to a smaller subspace. Descript audio codec (Kumar et al., 2023a) is a

model that brings a few improvements to the RVQ-GAN baselines of neural audio codec. Specifically, they show that projecting the bottleneck dimension to smaller subspaces before applying quantization scheme allows for improved codebook usage and greatly reduces the number of "dead codes" (codes that are never used by the audio codec). While they showcase improvements on the compression task, this approach tends to results in reduced modeling performances. In Snac (Siuzdak et al., 2024), the model uses a different granularity for different VQ streams. The model operates at a lower frame rate for the coarse tokens and has growing frame rate as the representation gets to the more detailed token streams. Related approaches such as VRVQ (Chae et al., 2025) use a variable level of granularity depending on the amount of information in the audio, resulting in a variable bitrate codec.

All previously exposed methods rely on what is referred to as **acoustic tokens** *i.e.* tokens whose only purpose is to reconstruct the audio signal back. Those are often opposed to **semantic tokens**, which aim to capture higher-level, meaning-related aspects of the audio, such as phonemes, words, or speaker identity, rather than low level signal information. These tokens are typically obtained by training models to learn discrete representations that encode semantic information, often using clustering or quantization techniques on the latent space of self-supervised models (see section 3.2). Recent methods use a blend of both tokens. SemantiCodec (Liu et al., 2024a) proposes an ultra low bitrate codec based on mel-spectrogram compression that uses both types of tokens. Mimi (Défossez et al., 2024) is a codec that uses a single encoder to extract acoustic tokens used for reconstruction and semantic tokens distilled from WavLM model (Chen et al., 2021b). While results do not indicate significant gains on compression, the inclusion of the semantic tokens seems to help the downstream language modeling task.

5.3 Audio Language Modeling

As described in the previous section, neural audio codecs are able to transform audio signals into sequences of discrete tokens. These tokens can then be used to train language models, similar to how text language models are trained. An audio language model is trained to predict the next token in a sequence of audio tokens, effectively learning the distribution of audio sequences. At inference the LM can be used to generate new sequences of audio tokens. Those generated tokens can be converted to audio signals

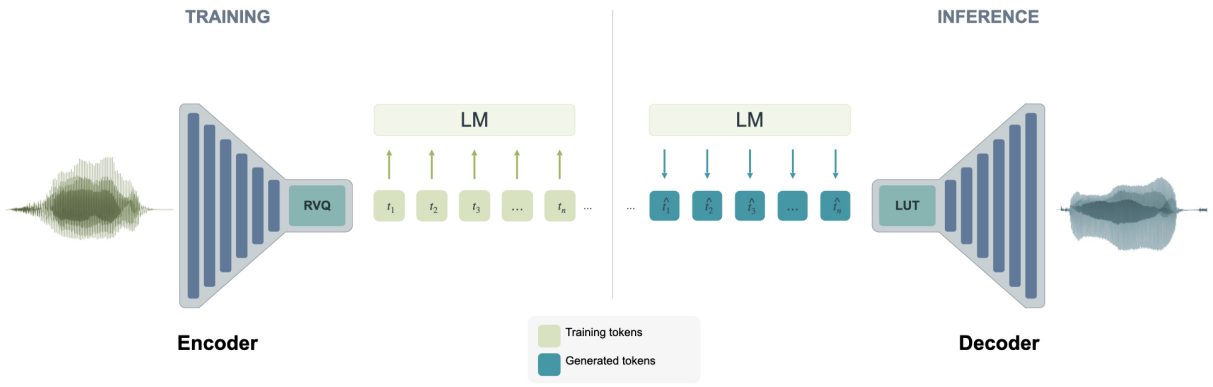

Figure 5.1 – Overview of the training and inference process of an audio language models. At training time audio signals are encoded into training tokens using the encoder of the neural audio codec in order to train the LM. At inference time, the LM generates autoregressively a sequence of tokens that are then decoded into audio using the decoder of the neural audio codec. The encoder and decoder depicted are the one from the Neural Audio Codec. RVQ and LUT (Look Up Table) respectively represent the processes of quantizing the embeddings into tokens and looking up in the codebook which vector corresponds to which token.

using the codec decoder. The training and inference process of the audio LM is depicted in Figure 5.1.

This section goes over details for the modeling of audio tokens using Audio Language models together with some examples of use cases and models using this approach.

5.3.1 Architecture

The architecture of audio language models is very similar to the architecture of text LLMs and is based on the Transformer architecture Vaswani et al. (2017). To adapt it to multiple streams of tokens some modifications are usually made. First, similar to the RVQ decoding, the LM has K learned embeddings tables and embeddings are summed to obtain the first sequence of vectors in the model. Second, since the model has to predict tokens in different vocabularies, the output of the models is usually replaced by K heads

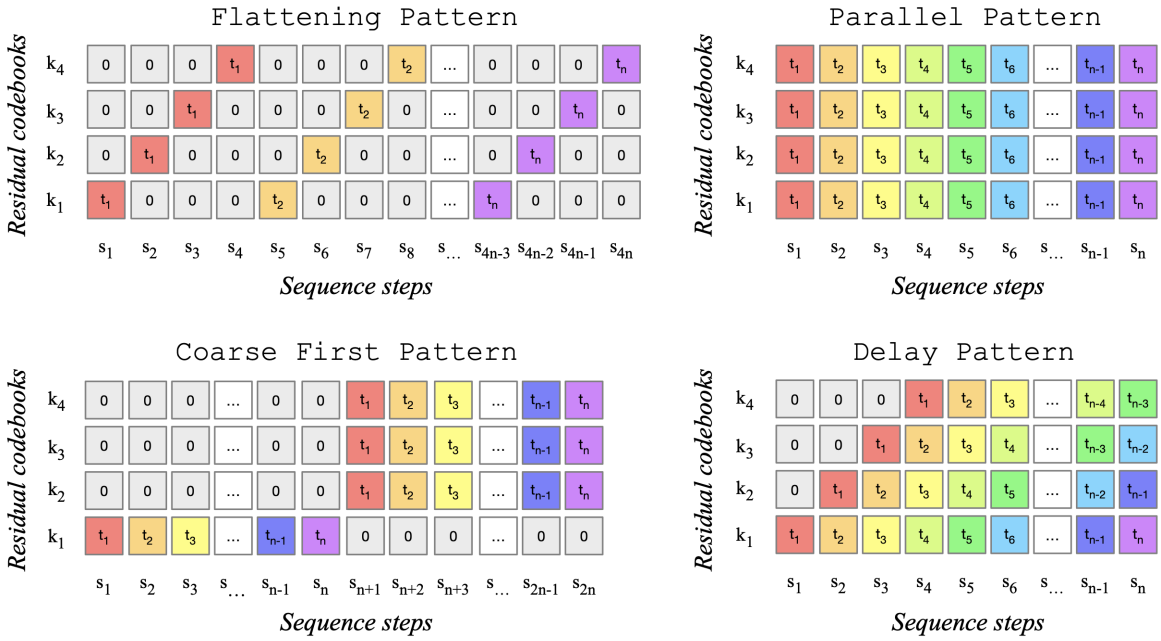

Figure 5.2 – Usual ways to handle streams in the literature, (source Copet et al. (2024))

each predicting their own vocabulary.

5.3.2 Handling streams of tokens

The first thing that is different from text language models is the fact that RVQ is giving streams of parallel tokens. One needs a way to input those tokens to the LM and also a way to make the language model output those tokens in a sensible way. We will go over the common strategies present in the literature. Figure 5.2 illustrates common strategies.

Flattening

The simplest way to handle the parallel streams of tokens is to flatten them into a single sequence. This process transforms K parallel sequences of length T_1 into a single $1-D$ sequence of size $K \times T_1$. Typically, in this case a conditioning k is given to the model with the input sequence which specifies from which vocabulary (*i.e.* VQ codebook) the next token is from.

When performing autoregressive generation, this way of flattening is problematic be-

cause the sequence's length is now K times larger. This means that one needs to call the model K times more to generate a sequence of length T_1, which is not efficient.

Parallel

One solution that keeps the length of the sequence to T_1 is to generate all the VQ tokens from a time step t in parallel. The idea is to have an LM with K heads all responsible for predicting in one VQ vocabulary. Although this parallelization makes processing more efficient by dividing the number of model pass by K, it poses a number of sampling issues because streams are treated as if they were independent from one another.

As an example, suppose the model predicts two equally likely futures: a guitar note or a trumpet note. Each corresponds to a consistent set of tokens across all VQ streams. If tokens are sampled in parallel and independently, the model might generate a guitar token in one stream and a trumpet token in another producing a mix that doesn't correspond to either coherent sound. This mismatch wouldn't happen with sequential sampling, which maintains consistency across streams.

Coarse-First

The idea of *coarse-first* usually involves two models Ho et al. (2022). The first one is generating the first VQ stream also called *coarse stream*. This will result in generating most of the semantic information from the audio. A second model will then generate the rest of the VQ streams in parallel conditioned on the coarse tokens. This model is basically generating the details, making the resulting audio high quality. This solution, while having still independent sampling issues, is robust to the issue described in the previous section. If the sampled coarse token is a guitar token, the second model will assign 0 probability to all the trumpet tokens in the subsequent VQ streams. This procedure is reducing the number of iterations to $2 * T_1$ this also allows for potentially having smaller models since the task has been divided into two simpler sub tasks.

Recently, Moshi (Défossez et al., 2024) have used a combination of a two-model strategy with the flattening approach. A first model is generating a continuous embedding for the next time step. Based on this embedding, another lighter model will autoregressively predict the VQ tokens. This approach can be seen as similar to the coarse first, one model

is taking care of the semantics of the audio while a second model is in charge of generating the acoustic realisation. The advantage of this approach is that it makes the model fully streamable. While you still need $K+1$ passes per time step, the size of the second model can be made significantly smaller than the first one.

Delay pattern

One strategy used in some works (Copet et al., 2024) allows to have the efficiency of parallel token generation, while having the consistency of sequential sampling. The idea is to introduce a delay in the VQ streams. The coarse stream is not delayed, while the first residual stream is delayed by one time step, the second residual stream is delayed by two time steps, and so on. Overall, the number of generations is reduced to $T_1 + K$ due to the delays. This approach is very efficient while not reducing the theoretical capabilities of the model.

5.4 Applications

While some models such as AudioLM (Borsos et al., 2023) or AudioGen (Kreuk et al., 2023) are general models trained on a variety of different audio modalities (speech, music, environmental sounds...), most models get the best performance by specialising to a specific type of audio. We will go over most important models in different audio modalities, namely speech, music and environmental sounds.

5.4.1 Speech Generation

As in audio research as a whole, speech processing has been a driving force of the field of audio language modeling. The field can be subdivided into two types of models: text-to-speech models and conversational models.

Text-to-Speech LM

The first work using Audio Language models for the task of text-to-speech (See Section 4.2.1) was Vall-E (Wang et al., 2023a). It was then followed be a series of follow up works such as Vall-E2 (Chen et al., 2024c) Vall-E-X (Zhang et al., 2023) etc.. Those models are trained on a large corpus of text and speech pairs. The conditioning is done by extracting phoneme sequence from the text and using them as conditioning for the model.

The model is also conditioned with an audio sample of the target speaker allowing the model to generate speech in the voice of a target speaker.

Recently, Kyutai introduced Unmute.sh a way to use Moshi (Défossez et al., 2024) to perform text-to-speech generation as well as ASR in a streaming way. It allows for using any, sufficiently fast, text language to become a real time speech conversational model.

While this type of systems allow to use the power of large language models, all the content that is not contained in the text is lost. Making the core semantic model unable to understand or generate speech elements that are not contained in the text such as emotions, intonations or other speech features.

5.4.2 Speech models

Learning to model speech from raw audio using language models started with models that rely principally on semantic tokens. GSLM (Lakhotia et al., 2021) use Hubert tokens (Hsu et al., 2021). In this work the authors managed to train a model capable of generating words and small sentences using solely audio without using any text in any of the training stages.

Obviously, leveraging text can drastically improve the performances of such speech models. Text is a very compressed and discrete way to semantically represent the content of a speech sample. Some models (Défossez et al., 2024; Hassid et al., 2023) started to incorporate speech tokens into already pretrained text LLMs. Using text tokens also helps those models to generate coherent speech outputs at a longer range than previously possible. SpiritLM (Nguyen et al., 2024) used interleaved speech and text tokens to allow the model to handle both text and speech either as input or output of the model.

Recently, full dialogue systems relying on audio LMs were designed. Moshi (Défossez et al., 2024) introduces a novel dual audio stream approach. One stream is dedicated to the output and one stream to the input. This allows the model to handle both input and output audio in a streaming way. The model is thus capable of listening and generating speech at the same time. This is a significant step towards building real-time conversational agents that can understand and generate speech in a natural way.

5.4.3 Music Generation

The field of Music Generation is one of the main fields that benefitted from Audio Language Models. Some startups such as Suno (2025) or Udio (2025) have already built music generation models that are used by thousands of users.

Text to Music Generation

The first type of conditioning that has been used for Music generation was text conditioning. Models such as MusicLM (Agostinelli et al., 2023) and MusicGen (Copet et al., 2024) built simple Audio Language models that would be conditioned on text either through simple T5 encoder when pairs of audio text conditioning are available or using Clip style Radford et al. (2021) text-audio joint embedding space Elizalde et al. (2023); Huang et al. (2022a). The embedding from the text is then fed to the model and processed using cross attention mechanism Bahdanau et al. (2015).

Other conditioning for Music Generation

Other types of conditioning have been used for music generation. For example, MusicGen (Copet et al., 2024) also introduced a conditioning based on a melody. This conditioning is based on a sequence of notes that can be used to guide the model in generating music that follows a specific melody. The melody is encoded using a chromagram representation that is then fed to the model as a conditioning.

Another type of conditioning that has been used is the conditioning on a style. MusicGen-Style (Rouard et al., 2024) is a model that uses a style conditioning based on a sequence of audio samples that are used to guide the model in generating music that follows the style of a specific audio sample.

5.5 Concurrent Approaches

While the previous sections focused on the use of autoregressive audio language models, there are other approaches that have been used to generate audio in the latent space of neural audio codecs.

Magnet (Ziv et al., 2024) or VampNet (García et al., 2023) propose a different approach based on masked language modeling instead of next token prediction. This type of

non auto regressive models result in a lower latency, but they tend to have comparatively weak performances.

Notably Diffusion and Flow Matching are techniques that have been used to generate long form audio. Similarly to audio language models, those models rely on the use of audio codecs to reduce the sequence length of the audio to a manageable size. But contrary to audio language models, they do not usually use the discrete but rather the continuous latent space of the neural audio codec (before the quantization). Those methods have been used in many setups such as TTS (Ju et al., 2024; Shen et al., 2024) voice cloning (Le et al., 2024), music generation (Chen et al., 2024b; Evans et al., 2024; Huang et al., 2023; Li et al., 2024) general purpose foundation models (Liu et al., 2023c; 2024b; Vyas et al., 2023).

Since neural audio codecs typically lose audio fidelity with the quantization step, diffusion and flow matching theoretically allow for higher quality audio generation. However in practice, if K is not too small, the quantization step is often not the bottleneck of quality of the audio generation process.

While it is unclear which of these approaches will prevail in the long term, all rely on the same principle: Generating audio in the latent space of a neural audio codec.

Chapter 6

Decoding Audio Tokens with Diffusion

This chapter describes the paper "From Discrete Tokens to High Fidelity Audio Using Multi-Band Diffusion" (San Roman et al., 2023) published at NeurIPS 2023. It focuses on the last part of the audio LM generation pipeline, decoding discrete audio tokens into waveform audio. It proposes a novel diffusion-based method that can be plugged as a slower but higher quality decoder on top of Encodec (Défossez et al., 2023). This approach allows to have a second decoding option that complements the fast but lower quality GAN based decoder.

6.1 Introduction

Neural vocoders have become the dominant approach for speech synthesis due to their ability to produce high-quality samples (Tan et al., 2021). These models are built upon recent advancements in neural network architectures such as WaveNet (Oord et al., 2016) and MelGAN (Kumar et al., 2019), and have shown impressive results in generating speech with natural-sounding intonation and timbre.

In parallel, SSL applied to speech and audio data (Hsu et al., 2021; van den Oord et al., 2019) have led to rich contextual representations that contain more than lexical

content, e.g., emotion and prosody information (Kharitonov et al., 2022a; Kreuk et al., 2022). Generating waveform audio from such representations is hence a new topic of interest (Huang et al., 2022b; Liu et al., 2019; Polyak et al., 2021b). This is often performed in a two stage training pipeline. First, learn audio representations using SSL objectives, then, decode the speech using GAN approach such as the HiFi GAN model (Kong et al., 2020). Even though these methods perform well, they are known to be unstable, difficult to train and prone to add audible artifacts to the output waveform.

Compression models (Défossez et al., 2023; Zeghidour et al., 2021) can also be considered as SSL models that use the reconstruction loss as a way to learn meaningful representations of the data. Unlike models described before, compression models are trained in an end-to-end fashion, while learning both audio representation (often a discrete one) and synthesis, and can model a large variety of audio domains. They are optimized using complex combinations of specifically engineered objectives including spectrogram and feature matching, as well as multiple adversarial losses (Défossez et al., 2023). Even though they have impressive performance compared to standard audio codecs, e.g., Opus (Valin et al., 2012), they tend to add noticeable artefacts when used at very low bit rates (e.g. metallic voices, distortion) that are often blatantly out of distribution.

After model optimization the learned representation can also be used for different audio modeling tasks. Kreuk et al. (2023) presented a textually guided general audio generation. Wang et al. (2023a) proposed a zero shot text to speech approach. Agostinelli et al. (2023) demonstrated how such representation can be used for text-to-music generation, while Hsu et al. (2022) followed a similar modeling approach for silent video to speech generation.

In this work, we present Multi-Band Diffusion (MBD), a novel diffusion-based method. The proposed approach can generate high-fidelity samples in the waveform domain of general audio, may it be speech, music, environmental sounds, etc. from discrete compressed representations. We evaluate the proposed approach considering both objective metrics and human studies. As we demonstrate empirically, such an approach can be applied to a wide variety of tasks and audio domains to replace the traditional GAN based decoders. The results indicate that the proposed method outperforms the evaluated baselines by a significant margin.

Our Contributions: We present a novel diffusion based model for general audio syn-

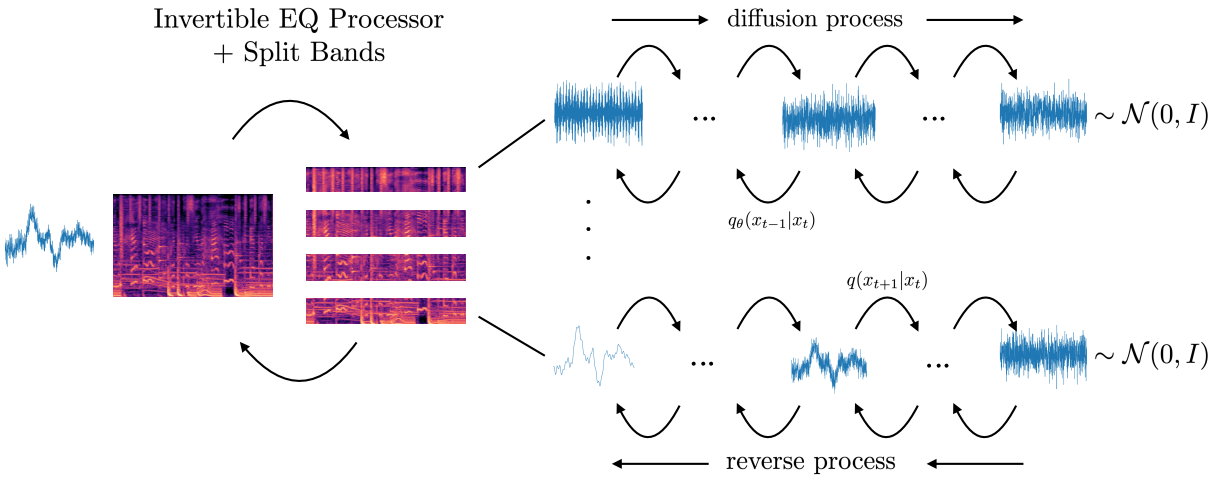

Figure 6.1 – Multi-Band Diffusion process (resp. reverse process). The first step consists of a reversible operation (EQ Processor) that normalizes the energy within frequency bands to resemble that of a standard Gaussian noise. The audio is then filtered into non-overlapping bands. Each band has its own diffusion process using a specifically tuned version of the proposed *power noise schedule*.

thesis. The proposed method is based on: (i) a **band-specific diffusion model** that independently processes different frequency bands, allowing for less accumulative entangled errors; (ii) a **frequency equalizer (EQ) processor** that reduces the discrepancy between the prior Gaussian distribution and the data distribution in different frequency bands; and (iii) A novel **power noise scheduler** designed for audio data with rich harmonic content. We conduct extensive evaluations considering both objective metrics and human study, demonstrating the efficiency of the proposed approach over state-of-the-art methods, considering both GAN and diffusion based approaches.

Most diffusion models that sample data from complex high dimensional distributions use upsampling frameworks (Huang et al., 2023; Takahashi et al., 2023). This type of cascaded models are achieving good performance but they are based on series of diffusion processes conditioned on the output of the previous and thus can not be performed in parallel. In vision, some efforts have been invested in simplifying diffusion pipelines. SimpleDiffusion (Hoogeboom et al., 2023) presents a framework that matches results of cascading diffusion models using a single model. The model architecture and training objective are adapted to focus on low-frequency content while keeping high quality textures.

To the best of our knowledge, this type of idea has not been ported to audio processing as of yet.

Finally, our work offers an alternative to the decoder of adversarial neural audio codecs such as SoundStream (Zeghidour et al., 2021) and EnCodec (Défossez et al., 2023), which consist in an encoder, a quantizer, and a decoder, and are trained with combination of losses including discriminators, spectrogram matching, feature matching, and waveform distances. Our diffusion based decoder is compatible, but offers higher quality generation as measured by subjective evaluations.

6.2 Method

6.2.1 Background

Following Ho et al. (2020), we consider a diffusion process given by a Markov chain q where Gaussian noise is gradually added to corrupt a clean data point x_0 until a random variable x_T close to the standard Gaussian noise is obtained. The probability of the full process is given by

$$q(x_{0:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}), \tag{6.1}$$

where $q(x_t|x_{t-1}) \sim \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$ and $(\beta_t)_{0\leq t\leq T}$ is usually referred to as the noise schedule. One can efficiently sample any step of the Markov chain t with

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon, \tag{6.2}$$

where $\bar{\alpha}_t = \prod_{s=0}^{t}(1-\beta_s)$ is called the noise level and $\varepsilon \sim \mathcal{N}(0, I)$. DDPM aims at going from prior noise x_T to the clean data point x_0 through the reverse process

$$p(x_{T:0}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \tag{6.3}$$

where $p_\theta(x_t|x_{t+1})$ is a learned distribution that reverses the diffusion chain $q(x_{t+1}|x_t)$ and $p(x_T)$ is the so-called *prior* distribution that is not learned. Under the ideal noise schedule, one can see from eq. (6.2) that the prior distribution can be approximated by $\mathcal{N}(0, I)$.

Ho et al. (2020) show that the distribution $p_\theta(x_{t-1}|x_t)$ can be expressed as $\mathcal{N}(\mu_\theta(x_t,t),\sigma_t I)$ where μ_θ can be reparameterized as follow:

$$\mu_\theta(x_t,t)=\frac{1}{\sqrt{1-\beta_t}}\left(x_t-\frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t,t)\right). \tag{6.4}$$

This reparametrization allows to train a neural network ε_θ to predict the noise in the corrupted data point x_t. To train this neural network, one can use the simple objective given by Ho et al. (2020) that consists in sampling x_t using eq. (6.2) and optimizing the following L2 loss:

$$\mathcal{L}=\mathbf{E}_{x_0\sim d(x_0),\varepsilon\sim\mathcal{N}(0,I),t\sim\mathcal{U}\{1,.,T\}}\left(||\varepsilon-\varepsilon_\theta(\sqrt{\bar{\alpha}_t}x_0+\sqrt{1-\bar{\alpha}_t}\varepsilon,t)||^2\right). \tag{6.5}$$

With such a model, one can reverse the diffusion process iteratively using the following equation:

$$x_{t-1}=\frac{1}{\sqrt{1-\beta_t}}\left(x_t-\frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t,t)\right)+\sqrt{\sigma_t}\varepsilon, \tag{6.6}$$

where σ is a parameter that should be chosen between $\tilde{\beta}_t=(1-\bar{\alpha}_{t-1})/(1-\bar{\alpha}_t)\beta_t$ and β_t (Ho et al., 2020). In our experiments we always use $\sigma_t=\tilde{\beta}_t$.

6.2.2 Multi-Band Diffusion

The Multi-Band Diffusion method is based on three main components: (i) Frequency EQ processor; (ii) Scheduler tuning; and (iii) Band-specific training, which we now describe.

Frequency Eq. Processor The mathematical theory of diffusion processes allows them to sample from any kind of distribution, regardless of its nature. However, in practice, training a diffusion network for multiple audio modalities in the waveform domain is an open challenge. We make the assumption that the balance of energy levels across different frequency bands in both the prior Gaussian distribution and the target distribution is important to obtain an efficient sampling mechanism.

A white Gaussian noise signal has equal energy over all frequencies. However natural sounds such as speech and music do not follow the same distribution (Schnupp et al., 2011), i.e. music signals tend to have similar energy level among frequency bands that are

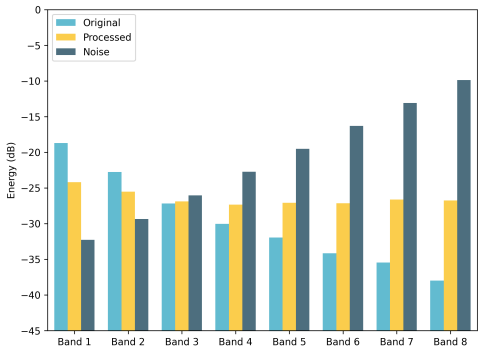

Figure 6.2 – Standard deviation in 8 mel scale frequency bands (from lows to highs). For data from our dataset (Original), Equalized data (Processed) and for standard Gaussian Noise (Noise).

exponentially larger. For signals of the same scale, white noise has overwhelmingly more energy in the high frequencies than common audio signals, especially at higher sample rate (see 6.2). Thus during the diffusion process, high frequency content will disappear sooner than the low frequency counterpart. Similarly, during the reverse process, the addition of noise given by equation 6.6 will have more impact over the high frequencies.

To resolve this issue, we normalize the energy of the clean signal, denoted as x_0, across multiple frequency bands. We split x_0 into B components b_i, with a cascade of band pass filters equally spaced in mel-scale. Given the filtered band b_i of an audio signal, we normalize as follow,

$$\hat{b}_i = b_i \cdot \left(\frac{\sigma_i^\epsilon}{\sigma_i^d} \right)^\rho , \tag{6.7}$$

where σ_i^ϵ and σ_i^d denote the energies in the band i for standard Gaussian noise and for the signals in the dataset, respectively. The parameter ρ controls to what extend we align the energy levels. For $\rho = 0$ the processor does not do any rebalancing and $\rho = 1$ corresponds to matching exactly the target energy. Given that speech signals often have no content in the high frequency bands, we compute the parameters σ_i^d over the music domain to avoid instabilities in equation 6.7.

Scheduler Tuning. The noise schedule is known to entail an important set of hyperparameters in diffusion models, and to play a critical role in the final quality of the generation (Karras et al., 2022).

A common approach when generating raw waveform is using either linear or cosine

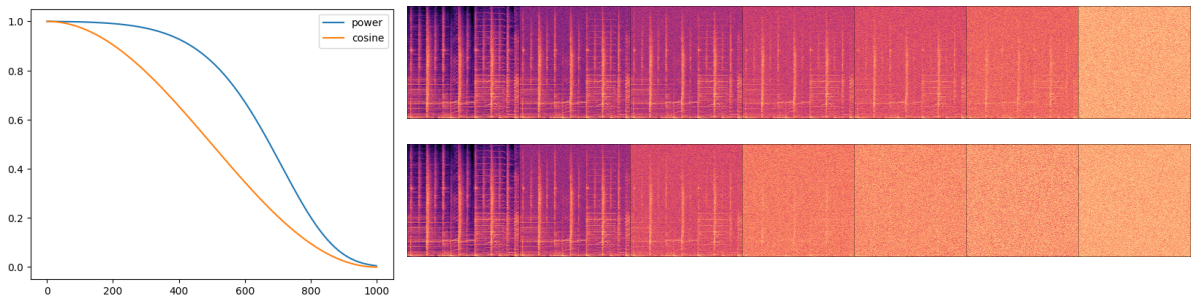

Figure 6.3 – Left curves depict a comparison of the noise level $\bar{\alpha}$ along the diffusion process for cosine schedule and our power schedule. Right figure presents spectrograms along diffusion process. The top row is our power schedule and the bottom row follows cosine schedule.

schedules (Nichol and Dhariwal, 2021). Such schedulers performs good for read speech where the frequency spectrum is not wide or for low-sampling-rate generation followed by cascaded models that iteratively upsample the signal. In preliminary experiments, we found such schedules performs poorly when generating signals at high sampling rate. Hence, we argue that one should prefer a more drastic schedule. We propose to use *p-power schedules*, defined as:

$$\beta_t = \left(\sqrt[p]{\beta_0} + \frac{t}{T}(\sqrt[p]{\beta_T} - \sqrt[p]{\beta_0}) \right)^p , \tag{6.8}$$

where the variance of the injected noise at the first and last step (β_0 and β_T) are hyperparameters. One could assume that, since the noise schedule used at generation time can be chosen after training, it is unnecessary to tune the training noise schedule and only focus on the choice of subset S. As evoked by Chen et al. (2021a), in practice, the training noise schedule is a crucial element of diffusion models. Since the train-time steps are sampled uniformly, the training noise schedule parameterizes the sampling distribution of the noise level $\sqrt{\bar{\alpha}}$. As seen in 6.3, using the proposed power schedule will results in sampling most of the training examples very small amount of noise (i.e. very high $\bar{\alpha}$).

We noticed that for a time step t close to T, i.e. at the end of the diffusion process, the model estimate $\epsilon_\theta(x_t)$ of the noise is often worse than simply using x_t itself. We hypothesize this is due to the limited precision when training the model. In that regime, the model can advantageously be replaced by the identity function, which is equivalent to skipping those timesteps entirely. We thus choose the β_t values such that $\sqrt{1-\alpha_t}$ is large enough to avoid this phenomenon.

Band-Specific Training. Similarly to the findings in image diffusion models (Song et al., 2021), audio diffusion models first generate low frequencies and then address high frequencies during the final stages of the reverse process. Unlike images where high frequencies are only locally connected, audio data contains complex entanglements of spectrogram content across both time and frequency (Schnupp et al., 2011). As a result, training a diffusion model on full-band audio data would always provide the ground truth low frequencies when generating high frequencies. It ends up amplifying the errors committed at the beginning of the generation when unrolling the reverse process.

Following that, we proposed training each frequency band independently, denoted as Multi-Band Diffusion. Through preliminary experiments we found such an approach resulted in significant improvements in the perceptual quality of the samples. Interestingly, dividing the frequency band along model channels did not yield the same quality improvements. This observation supports our intuition that, by not providing the model with previously generated content (lower frequencies) during training, the model can avoid accumulating errors during sampling.

6.3 Experimental Setup

6.3.1 Model & Hyperparameters

Overview. Our approach serves as a replacement for EnCodec's decoder. This approach offers the advantage of flexibility and compatibility. It allows one to switch between the original decoder and the proposed diffusion decoder depending on the required trade-off between quality and generation time.

Architecture. Similarly to Chen et al. (2021a); Kong et al. (2021); Lee et al. (2022), we use a fully convolutional symmetrical U-net network (Ronneberger et al., 2015) with an alternation of two residual blocks (Défossez et al., 2021) and downsampling (resp. upsampling in the decoder) convolutional blocks of stride 4. The input audio conditioning is incorporated in the bottleneck of the network whereas the timestep t is embedded through a learned lookup table and added at every layer. According to the recommendation from Hoogeboom et al. (2023), it is advisable to allocate additional computational resources close to the bottleneck of the network when applying diffusion to high-dimensional data. Hence, we opted for a growth rate of 4. The weight of one model is 1 GB. A visual

description of the model architecture can be seen in Figure 6.4.

Input Conditioning. We use the latent representation of the publicly available EnCodec models at 24kHz (Défossez et al., 2023) which are frozen during training. The embedding sequence is upsampled using linear interpolation to match the dimension of the UNet bottleneck. In the experiments we include reconstructions using 1, 2 and 4 for EnCodec code books which correspond to bit rates of respectively 1.5kbps, 3kbps and 6kbps, when using multiple code books the embedding used is simply the average of the different code books.

Schedule. We trained our diffusion models using our proposed power schedule with power $p = 7.5$, $\beta_0 = 1.0\text{e}{-5}$ and $\beta_T = 2.9\text{e}{-2}$. Although we use very few diffusion steps (20) at generation time, we observed that it is beneficial to use many steps at training time (1000). First, it increases the versatility of the model since one can sample using any subset of steps $S \subseteq \{1, \dots 1000\}$. Second, it allows the model to be trained on a more diverse range of noise levels $\sqrt{\bar{\alpha}_t}$. In the experiment section we always use the simplest time steps sub sampling i.e. $S = \{i * \frac{1000}{N}, i \in \{0, 1, ..., N\}\}$ where N is the number of sampling steps (20 if not precised).

Frequency EQ processor. In the experiments we use a band processor that uses 8 mel scale frequency bands with $\rho = 0.4$. We compute the values of the bands σ_i^d on an internal music dataset.

Band Splitting. As described in 6.2 we use separate diffusion processes. In this work we always use a split of 4 frequency bands equally space in mel-scale using `julius`[1] Those bands are not related to the processor bands. The 4 models share the same hyperparameters and schedule. All models take the same EnCodec tokens as conditioning input.

Training. We train our models using Adam optimizer with batch size 128 and a learning rate of 1e-4. It takes around 2 days on 4 Nvidia V100 with 16 GB to train one of the 4 models.

Computational cost and model size. Diffusion model sampling has an intrinsic cost that is due to the number of model passes that are required for generation. We provide in Table 6.7 the details for time consumption and number of parameters of Multi-Band

1. https://github.com/adefossez/julius

DIFFUSION.

6.3.2 Datasets

We train on a diverse set of domains and data. We use speech from the train set of Common Voice 7.0 (9096 hours) (Ardila et al., 2020) together with the DNS challenge 4 (2425 hours) (Dubey et al., 2022). For music, we use the MTG-Jamendo dataset (919h) (Bogdanov et al., 2019). For the environmental sound we use FSD50K (108 hours) (Fonseca et al., 2021) and AudioSet (4989 hours) (Gemmeke et al., 2017a). We used AudioSet only for the research that is described in the publication and for the benefit of replicability. For evaluation, we also use samples from an internal music dataset.

6.3.3 Evaluation Metrics

Human evaluation. For the human study we follow the MUSHRA protocol (Series, 2014b), using a hidden reference and a low anchor. Annotators were recruited using a crowd-sourcing platform, in which they were asked to rate the perceptual quality of the provided samples in a range between 1 to 100. We randomly select 50 samples of 5 seconds from each category of the the test set and force at least 10 annotations per samples. To filter noisy annotations and outliers we remove annotators who rate the reference recordings less then 90 in at least 20% of the cases, or rate the low-anchor recording above 80 more than 50% of the time.

Objective metrics. We use two automatic evaluation functions. The first one is the standard ViSQOL (Chinen et al., 2020) metric. [2]. The second one is a novel metric we introduce to measure the fidelity of the mel-spectrogram of the reconstructed signal compared with the ground truth across multiple frequency bands. Let us take a reference waveform signal $x \in \mathbb{R}^T$ and a reconstructed signal $\hat{x} \in \mathbb{R}^T$. We normalize x to have unit variance, and use the same scaling factor for $\hat{x}$. We take the mel-spectrogram of both, computed over the power spectrum with M mels, and a hop-length H, e.g.,

$$z = \text{mel}\left[\frac{x}{\epsilon + \sqrt{\langle x^2 \rangle}}\right], \qquad \text{and} \qquad \hat{z} = \text{mel}\left[\frac{\hat{x}}{\epsilon + \sqrt{\langle x^2 \rangle}}\right], \tag{6.9}$$

with $z, \hat{z} \in \mathbb{R}^{F \times T/H}$. We compute the mel-spectrogram distortion $\delta = z - \hat{z}$. Finally

2. We compute visqol with: https://github.com/google/visqol using the recommended recipes.

for each time step t and frequency bin f, we can compute a Signal-To-Noise ratio. In order to avoid numerical instabilities, and also not let the metric be overly impacted by near zero values in the ground truth mel-spectrogram, we clamp the SNR value between $-25dB, +25dB$, considering that any distortion lower than -25 dB would have a limited impact on perception, and that beyond +25 dB, all distortions would be equally bad. Indeed, due to the limited precision used in the computation and training of a neural network, it is virtually impossible to output a perfectly null level of energy in any given frequency band, although such empty bands could happen in real signals. Finally we get,

$$s = \text{clamp}\left[10 \cdot (\log_{10}(z) - \log_{10}(\delta))., -25\text{dB}, +25\text{dB}\right]. \tag{6.10}$$

We then average over the time steps, and split the mel-scale bands into 3 equally spaced in mel-scale. We report each band as Mel-SNR-L (low frequencies), Mel-SNR-M (mid frequencies), and Mel-SNR-H (high frequencies). Finally we also report the average over all 3 bands as Mel-SNR-A. At 24 kHz, we use a STFT over frames of 512 samples, with a hop length $H = 128$ and $N = 80$ mel bands.

6.4 Results

6.4.1 Multi modalities model

We first evaluate the performance of our diffusion method compared with EnCodec on a compression task. Specifically we extract audio tokens from audio samples using the EnCodec encoder and decode them using Multi-Band Diffusion and the original decoder.

We perform subjective evaluation on four subsets: 50 samples of clean speech from DNS, 50 samples of corrupted speech using DNS blended with samples from FSD50K, 50 music samples from Jamendo and 50 music samples from an internal music dataset. All speech samples are reverberated with probability 0.2 using room impulse responses provided in the DNS challenge. In Table 6.1, we present 3 subjective studies with different bit rate levels: 6 kbps, 3 kbps, and 1.5 kbps. Note that scores should not be compared across the studies since ratings are done relatively to the other samples of the study. We include Opus (Valin et al., 2012) at 6 kbps as a low anchor and the ground truth samples. Even though the comparison with EnCodec is done with different model sizes cf Table

Table 6.1 – Human evaluations (MUSHRA) scores for 24kHz audio. The mean and CI95 results are reported. The Opus low anchor and ground truth samples are consistent across all three studies, delimited by horizontal lines. The other methods used a bit rate of 6 kbps for the top study on top, 3 kbps for the middle one, and 1.5 kbps for the bottom one. Higher scores indicate superior quality.

Method	Speech	Music	Average
Reference	93.86±0.014	92.93±0.021	93.40
Opus	61.14±0.094	34.24±0.147	47.69
EnCodec	79.03±0.053	**84.67±0.062**	81.85
MBD (ours)	**84.68±0.036**	83.61±0.072	**84.15**
Reference	93.17±0.015	94.45±0.014	93.81
Opus	62.83±0.14	36.17±0.12	49.5
EnCodec	78.51±0.078	85.55±0.045	82.03
MBD (ours)	**84.42±0.042**	**87.31±0.041**	**85.87**
Reference	94.65±0.012	94.71±0.012	94.78
Opus	44.65±0.057	38.33±0.081	41.49
EnCodec	49.51±0.072	**75.98±0.077**	62.75
MBD (ours)	**65.83±0.056**	75.29±0.076	**70.56**

6.7, original paper (Défossez et al., 2023) makes it clear that the number of parameters of the model is not a limiting factor of their method.

Multi-Band Diffusion outperform EnCodec on speech compression by a significant margin, up to 30% better, while being on part on music data. Averaging across modalities, our method outperforms EnCodec for all bit rates. Qualitatively, we observed that GAN-based methods have a tendency to introduce very sharp and straight harmonics that can lead to metallic artifacts. On the other hand, our diffusion method produces more blurred high-frequency content. We provide a number of spectrogram in the Supplementary Material, Section 6.4.5.

In table 6.2, we compare our approach with other decoders baseline trained using the same condition and data as our model. Specifically we compare to HifiGAN (Kong et al., 2020) and PriorGrad (Lee et al., 2022) using the hyper parameters proposed on their original papers.

The second part of table 6.2 adds comparisons to other end to end audio codecs that do not rely on EnCodec. Specifically it adds the pretrained model of DAC (Kumar et al., 2023b) at 6 kpbs which is a different audio codec at 24khz. We show EnCodec +

Table 6.2 – Human evaluations (MUSHRA) scores for 24kHz audio. The mean and CI95 results are reported. The first part of the table reports different methods of EnCodec tokens at 6 kbps decoding while the second part adds other independent compression baselines at 6 kbps.

Method	score
Ground Truth	90.32 ±1.39
MBD	**85.16** ±0.93
Encodec	82.73±1.11
PriorGrad	65.16±2.2
HifiGan	82.5±1.25
DAC	84.44±1.14
OPUS	65±2.43

Multi-Band Diffusion is on part with DAC that uses a different quantized space. It is likely that training our Multi-Band Diffusion on the audio tokens of DAC would results in even higher audio quality.

6.4.2 Ablations

In this section we use objective metrics to compare the reconstruction performances. We compute for every experiment the ViQOL score and Mel-SNR on 3 mel spec bands. Objective metrics are computed on the same 4 types of modalities as in section 6.4.1 using 150 samples per category. Even though those metrics seem to not correlate well with human evaluation across different model families (c.f. Tables 6.1 and 6.3) in our testing it was accurately measuring the improvements in quality resulting from small design changes. In Table 6.3, we compare the reconstruction performances of Multi-Band Diffusion and Encodec at different bit rates. It is notable that overall Encodec achieves better objective reconstruction metrics while being outperformed in subjective evaluations. We argue that such models are better in spectral distance metrics due to their specific training for content reconstruction. On the other hand, diffusion based methods do not use feature or spectrogram matching and tend to create samples that are more "in distribution" resulting in more natural audio. Diffusion based methods have more freedom to generate something that will be different from the original audio. They are optimized to keep maximize likelihood of their output with respect to the train dataset. The optimal method might be different depending on the purpose. However we claim that

Table 6.3 – Objective and subjective metrics comparing the reconstruction performances of our model and EnCodec across bit rates.

Setting	ViSQOL (↑)	Mel-SNR-L (↑)	Mel-SNR-M (↑)	Mel-SNR-H (↑)	Mel-SNR-A (↑)
MBD @1.5 kbps	3.20 ±0.02	10.09	8 .03	8.26	8.79
EnCodec @1.5 kbps	3.33±0.02	9.61	10.8	13.37	11.36
MBD 3.0 kbps	3.47±0.02	11.65	8.91	8.69	9.75
EnCodec @3.0 kbps	3.64±0.02	11.42	11.97	14.34	12.55
MBD @6.0 kbps	3.67±0.02	13.33	9.85	9.26	10.81
EnCodec @6.0 kbps	3.92±0.02	13.19	12.91	15.21	13.75

Table 6.4 – Comparing the reconstruction performances of our model at 6 kbps.

Setting	ViSQOL (↑)	Mel-SNR-L (↑)	Mel-SNR-M (↑)	Mel-SNR-H (↑)	Mel-SNR-A (↑)
MBD @6.0 kbps	**3.67**±0.02	**13.33**	**9.85**	9.26	**10.81**
w-o Processor	3.38±0.02	13.16	9.68	8.46	10.43
Linear Schedule	2.93±0.03	10.65	7.10	7.73	8.49
Cosine Schedule	3.29±0.03	12.88	9.60	**9.59**	10.69
Single Band	3.32±0.02	12.76	9.82	8.58	10.39

our Multi-Band Diffusion is preferable for most generative tasks based on generation in the codec space.

To evaluate the impact of our individual contributions we performed an ablation study that evaluates models in the exact same setting when removing one element introduced in this chapter.

According to the findings of our study, increasing the number of steps to 20 results in improved output quality. However, further increasing the number of steps shows diminishing returns (Table 6.4). In comparison to our approach utilizing four models, a single model performs less effectively. Despite employing a similar number of neural function estimations it has worse audio quality and worse scores in every metrics. By leveraging our processor to rebalance the frequency bands, we achieved a notable enhancement of 0.2 in ViSQOL scores. Additionally, our proposed schedule demonstrates a performance increase of 0.4 and 0.2 when compared to standard linear and cosine schedules (Nichol and Dhariwal, 2021). Moreover, our proposed data processing technique also leads to a 0.2 increase in ViSQOL scores. The figures displayed in table 6.3 indicate that the high frequencies (Mel-SNR-H) are primarily affected by this processing technique.

6.4.3 Text to audio

Although our model alone cannot generate audio without conditioning, we show that when combined with a generative language model on the audio tokens, it provides substantial quality enhancements.

Text to Speech. Using language models on audio codecs has recently gained interest for Text to Speech. Methods such as VALL-E Wang et al. (2023a) or SPEAR-TSS Kharitonov et al. (2023a) achieved convincing results on this task. We claim that one can improve the quality of the final audio by just switching to our Multi-Band Diffusion token decoder To test that claim we use the implementation and pretrained models from Bark [3] that are publicly available. Bark is composed of three transformer models. The initial model converts text input into high-level self-supervised audio tokens, while the second and third models sequentially process these tokens to produce Encodec tokens with two and eight codebooks, respectively. We used our trained diffusion models to decode the final token sequences. We generated 50 text prompts from Bark in all supported languages. We also include 50 prompts using the music note emoji as suggested in the official Github page to generate some singing voices. We removed from the subjective tests the samples for which the language model failed to generate any voice, in our experiments using pretrained bark this happened for less than 5% of speech prompts and around 30% singing voice prompts. Table 6.5 presents the results, and we include the Encodec generation used in the original code base as a baseline.

Text to Music. There has been a significant advancement in the field of music generation using language modeling of audio tokens. Recently, this progress has been exemplified by MusicLM (Agostinelli et al., 2023) and MusicGen (Copet et al., 2024), which have greatly improved text-to-music generation. In order to demonstrate the versatility of our decoding approach, we utilized the open source version of MusicGen and trained a diffusion model conditioned with the tokens produced by its compression model. Our model is trained on the same dataset as the EnCodec model used by MusicGen, with a sampling rate of 32kHz. Additionally, we match the standard deviation of 16 mel scaled bands with the compression model output.

Notably, our method achieved a MUSHRA score improvement of +4 compared to standard MusicGen (see Table 6.5). Overall, the artifacts generated by the diffusion

3. https://github.com/suno-ai/bark

Table 6.5 – Human evaluations (MUSHRA) decoding token sequences from various methods.

Method	Bark			MusicGen
	Speech	Singing Voices	Average	Music
EnCodec	64.34±3.6	61.85±4.2	63.10	70.99±1.19
MBD	**76.04**±**2.9**	**73.67**±**3.4**	**73.86**	**74.97**±1.94

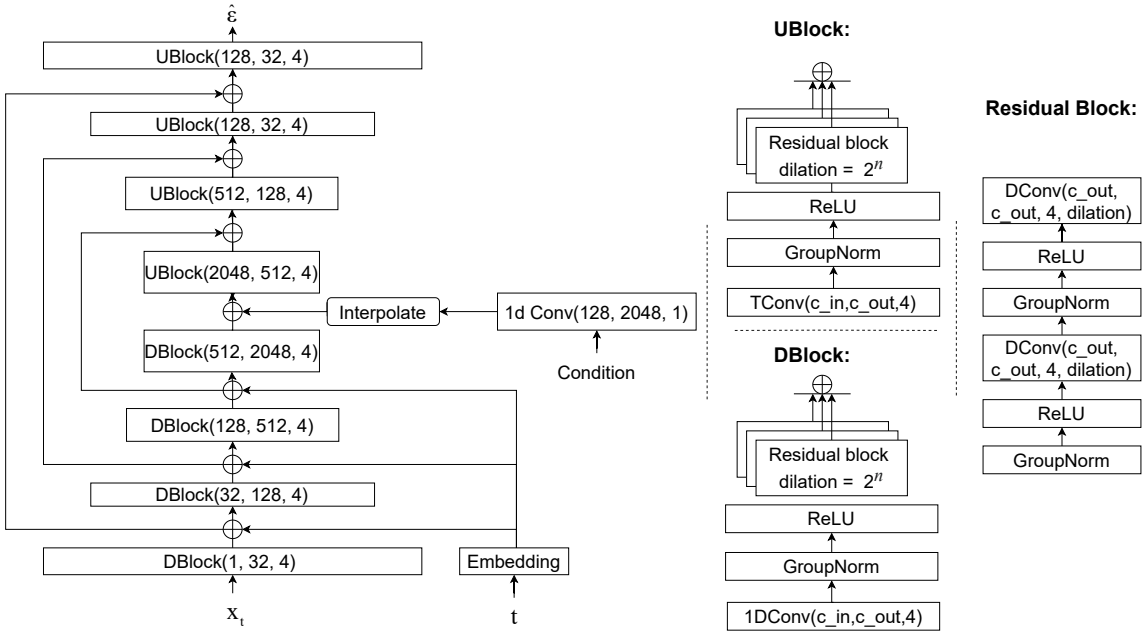

Figure 6.4 – The architecture of our diffusion model.

decoder are less pronounced. We find that in music containing complex elements, such as fast drum playing, the outputs from Multi-Band Diffusion are much clearer than the original ones.

6.4.4 Model

See Figure 6.4 for a detailed view of the model.

6.4.5 Additional Results

See Figures 6.5 and 6.6 for the spectrogram of the generated audio on a few examples. Finally, Table 6.6 studies the impact of the number of diffusion steps.

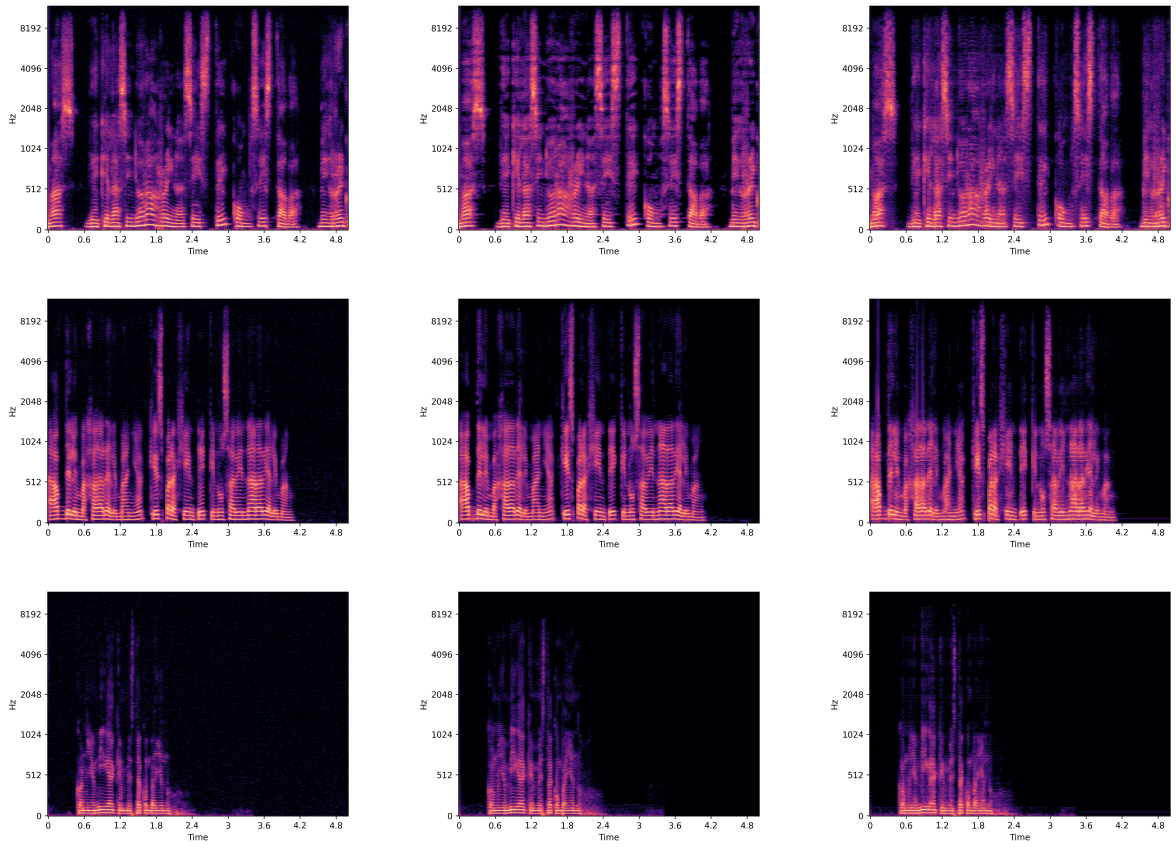

Figure 6.5 – Example of mel spectrograms from speech samples. (a) are generated with Multi Band diffusion, (b) are the ground truths and (c) are generated with Encodec.

Table 6.6 – Comparing the reconstruction performances of our model across number of steps at 6 kbps.

Setting	ViSQOL (↑)	Mel-SNR-L (↑)	Mel-SNR-M (↑)	Mel-SNR-H (↑)	Mel-SNR-A (↑)
MBD @6.0 kbps	3.67±0.02	13.33	9.85	9.26	10.81
500 steps	3.70±0.03	13.37	9.74	9.55	10.89
100 steps	3.64±0.03	13.36	9.90	9.42	10.89
10 steps	3.61±0.02	13.40	8.82	8.57	10.26

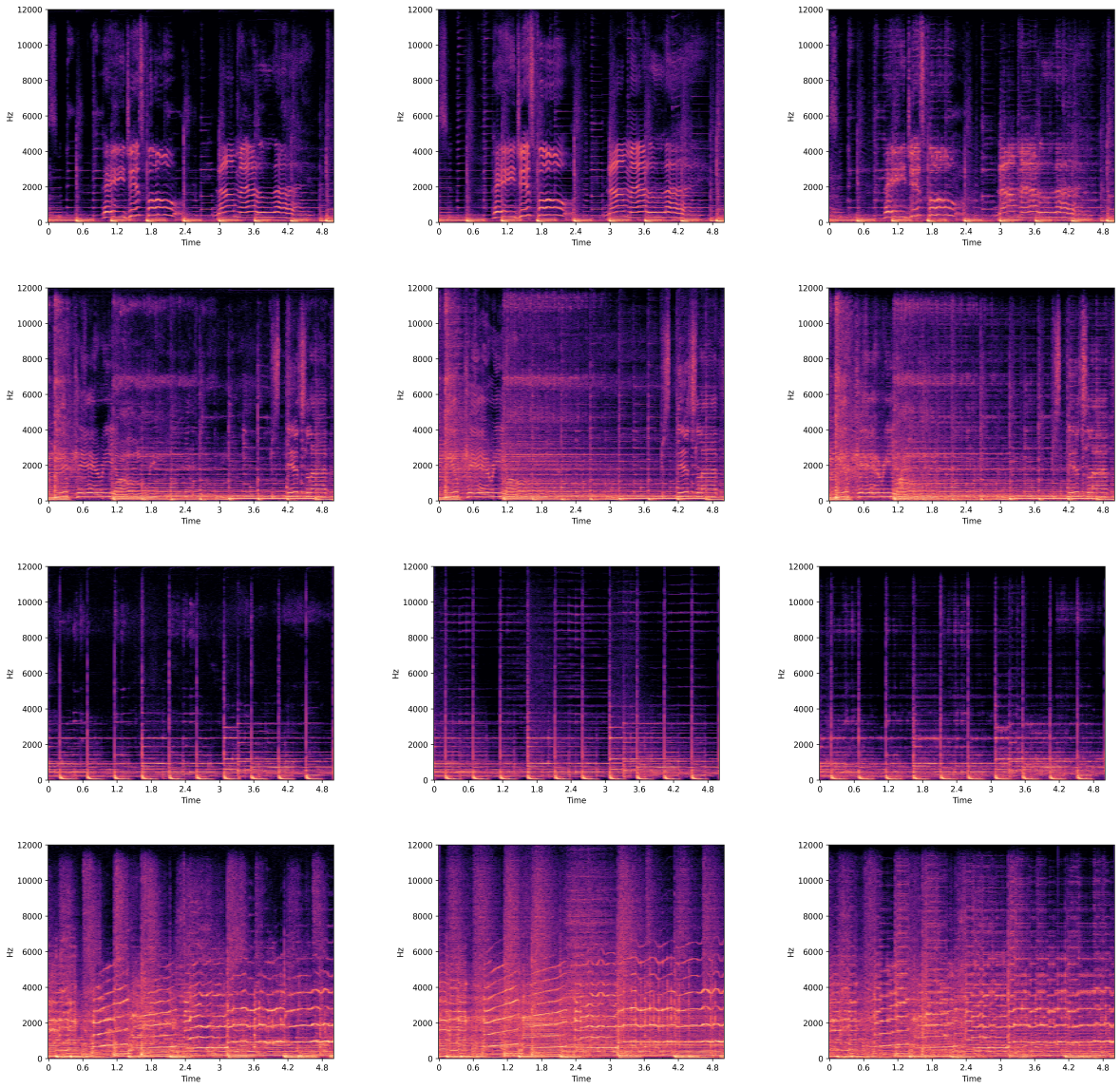

Figure 6.6 – Example of spectrograms from music sample. (a) are generated with Multi Band diffusion, (b) are the ground truths and (c) are generated with Encodec. In the first two rows, one can see examples of much sharper spectrograms, especially in the lower frequencies, from our reconstruction method compared to Encodec. In the third row one can denote that some of the high frequency spectral content can be less precise and sometimes blurred when using Multi Band diffusion. The last row is a sample where pitch smoothly changes, compared to our method Encodec fails to properly resynthesize it.

Table 6.7 – Table of computational time of MultiBand Diffusion and application for 30 seconds generations.

	Compute time (30s)	#parameters
Encodec	0.1s	56M
MBD	21.2s	411M
MusicGen-large + Encodec	102s	3.3B
MusicGen-large + MBD	123s	3.7B

6.5 Conclusion

In summary, our proposed diffusion-based method for decoding the latent space of compression models offers significant improvements in audio quality compared to standard decoders. While it does require more compute and is slower, our results demonstrate that the trade-off is well worth it. Our approach generates audio that is more natural and in distribution, with fewer artefacts compared to existing methods. However, it is worth noting that our method may not be suitable for all use cases. For instance, if real-time performance is a critical factor, our approach may not be ideal.

Ethical concerns. Our approach, although not categorized as generative AI, can seamlessly integrate with techniques like Wang et al. (2023a) to enhance the authenticity of generated voices. This advancement opens up potential misuses such as creating remarkably realistic deep fakes and voice phishing. Similar to all deep learning algorithms, our method depends on the quality and quantity of training data. We meticulously train our model on a substantial dataset to optimize its performance across a wide range of scenarios. Nevertheless, we acknowledge that imbalances in the dataset can potentially introduce biases that may impact minority groups.

Chapter 7

Stem based Music Generation

This chapter describes "MusicGen-Stem: Multi-stem music generation and edition through autoregressive modeling" a paper published at ICASSP 2025. It proposes to change the tokenization of MusicGen (Copet et al., 2024) to a stem-based approach. This change allows to build a model able to both generate and edit music stems.

7.1 Introduction

As described in section 5.4.3 Audio Language Model have been extensively used in the literature for the task of Music Generation. Those models achieve very good performances and manage to generate long and coherent audio sequences. However recent approaches have explored rich condition to control the music generation process (Tal et al., 2024; Wu et al., 2024b) most model are limited to the use of text-only conditioning. While these text prompts have the benefit of allowing extensive control over high-level musical characteristics of the generated music (like style, instrumentation, or mood), they remain limited with respect to precise control, specifically editing. For example, the production of a drum track for a given music piece can not be achieved using only a textual description. Additionally, the artist and music producers tend to have an iterative approach to music creation which is not something that is allowed by typical text-to-music models.

These limitations have led to research activities that aim extending the use cases

for music generation models. One line of research addresses the generation of musically coherent stems (or tracks) conditioned on audio input (Donahue et al., 2023; Mariani et al., 2024; Parker et al., 2024c; Yao et al., 2023a). If successful, these approaches would provide innovative means for the generation of musical accompaniment, for the iterative, stem-wise creation of a musical piece, and in combination with source separation, may allow replacing a stem in a given musical piece.

In line with these recent research activities this chapter introduces MusicGen-Stem, an extension of (Copet et al., 2024) towards multi-stem music generation that is able to perform at the same time text-to-music generation, text and audio conditioned stem generation, as well as iterative stem-by-stem generation. One of the main problems for training music generation for individual stems is the training data. Here we follow (Donahue et al., 2023) and use one of the state-of-the-art music source separation models (Défossez et al., 2021) to produce *bass*, *drum* and *other* stems. Note that we intentionally exclude vocal stems from the present study, on one hand to avoid the complexity of the generation of coherent lyrics, and on the other hand due to legal constraints.

MusicGen-Stem use cases

The proposed methods allow for several use-cases including:

— **Generate music given a textual prompt** and directly obtain 3 separated stems (*bass*, *drums* and *other*)

— **Generate the complementary stems** (e.g. add the *drums* and *other* instruments on top of the *bass*) given one or multiple stems (e.g. a *bass*). Here again the generation may be controlled by means of an additional text prompt

— Remove and **regenerate one or more of the three stems** in an existing song

— **Modify the sound texture of the *other* stem** by regenerating its RVQ residuals while keeping its first stream fixed (see 7.3.3).

Contributions

The contributions are as follows:

— MusicGen-Stem, a variant of the autoregressive text-to-music model MusicGen (Copet et al., 2024) that allows generating the three different stems that are used

in the present study. The proposed method can generate all stems at once, or individual stems conditioned on a given musical sample.

— To prevent the possibility of cross-talk across the stems, we propose to use specialized compression models that are used to tokenize the individual stems

— We evaluate on the text-to-music task and despite the additional complexity of the parallel generation of multiple stems, the proposed model is on par with its predecessor(Copet et al., 2024). Additionally, on the unconditional generation task, it outperforms all the previous multi-stem generative models on objective and subjective metrics

— We introduce a particular conditioning approach that allows our model to perform stem editing (replace an existing stem) and stem by stem generation. Our evaluations show that our approach compares favorably to the previous open-source models that have been proposed for the task of stem editing.

7.2 Related Work

7.2.1 Music editing models

To meet standard artist use cases other models have proposed more diverse ways of conditioning music generation models.

Conditional Music Generation

Wu et al. (2024a) propose Music ControlNet that uses a diffusion-based adapter that incorporates time-varying control signals (melody, dynamics, and rhythm) into a pre-trained text-to-music model using a pixel-wise conditioning mechanism on audio spectrograms. This architecture is useful for providing musicians with precise, localized control over specific musical elements, allowing for tasks like melody-based re-orchestration and precise volume shaping that standard text-to-audio models cannot achieve.

MusicGen (Copet et al., 2024) introduces a melody conditioning using Chroma feature. This allows to generates song with the same melody in very different genras. MusicGen-Style (Rouard et al., 2024), introduces a way condition music generation on other audio clips to make a new song in the same style.

Techniques of **zero-shot editing** such as DDPM inversion (Manor and Michaeli,

2024; Novack et al., 2024) and DDIM inversion (Zhang et al., 2024b) illustrate the potential of diffusion models to provide more flexibility and control in music generation. However, when attempting fine editing of a single instrument from a song, those approaches struggle to keep the rest of the track unchanged. This indicates that single stem models are not suited to the needs of real world artists.

Test-time optimization

Test-time optimization methods as well do not require training a model from scratch. For instance, textual inversion has been applied for diffusion (Plitsis et al., 2024) and autoregressive (Rouard et al., 2024) models, where given a pretrained frozen text-to-music model and a batch of audio that share similarities (e.g. same artist, style or instruments), a "pseudoword" in the text embedding space representing these similarities is obtained by doing a gradient descent on the "pseudoword" by optimizing the loss of the model. Still, these inversion methods often result in artifacts in the generated audio.

However, discrete models using quantized autoencoders lack flexibility for editing. While autoregressive models can extend a song by using its beginning as a prompt, they cannot perform inpainting or stem editing—essential for modifying specific sections without changing the entire piece.

To enable an autoregressive model to perform editing, instruction tuning can be done. For instance, in Instruct-MusicGen (Zhang et al., 2024a), the authors fine-tune a pretrained MusicGen model with a source separation dataset and instructions in order to perform adding, removing, extracting and replacing instruments. This method is limited to 5 seconds generation and we observe that the task of stem editing often fails to keep the remaining stems unchanged because of the non separation of the instruments in the streams of the model.

7.2.2 Multi-stem music generation and editing

In SingSong (Donahue et al., 2023), the authors perform vocal to accompaniment generation by conditioning an autoregressive MusicLM (Agostinelli et al., 2023) model with the coarse tokens of the compressed vocals stem. In StemGen (Parker et al., 2024c), the authors sequentially generate music stem by stem with a masked model transformer. The drawbacks of their model is that it needs a first stem as an input and then generates

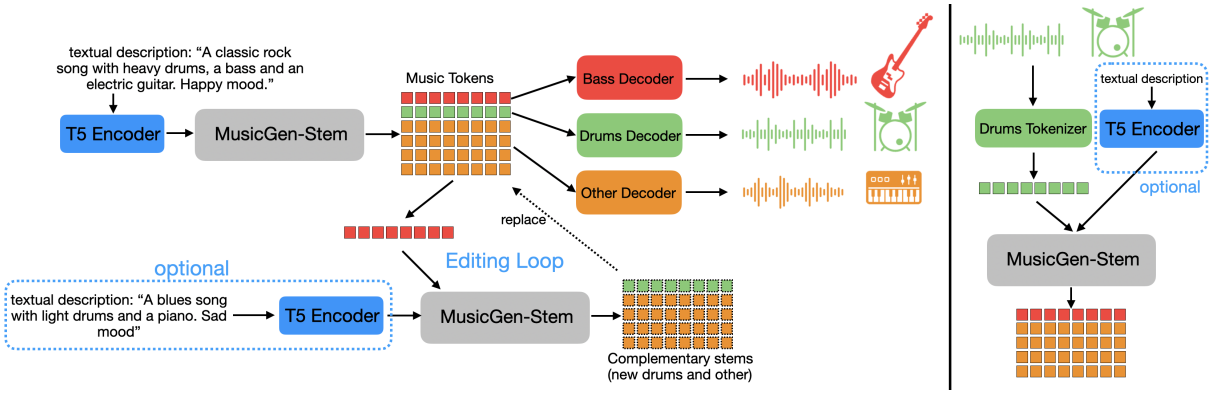

Figure 7.1 – Three use-cases of our model: **(up)** MusicGen-Stem can perform text-to-music generation and generates parallel streams of tokens representing the 3 stems (*bass, drums and other*). **(down)** MusicGen-Stem can also perform stem editing: given a subgroup of stems, the model can generate the complementary ones with an optional text prompt. **(right)** Given the waveform of one or multiple stems (that can be extracted from an existing song with Demucs), we tokenize them and MusicGen-Stem can generate the missing stems with an optional text prompt. We can then decode them.

music stem by stem which is compute intensive. In Jen-1 Composer (Yao et al., 2023b) as well as in Multi-source diffusion models (Mariani et al., 2024) (MSDM), the authors use a source separated dataset to train a diffusion model that outputs 4 stems in parallel. Jen-1 composer is a latent diffusion model whereas the second one is a diffusion model in the waveform space. They both can perform editing (i.e. replacing a stem in a song). Out of all of these models, MSDM (Mariani et al., 2024) is the only open source one. Karchkhadze et al.(Karchkhadze et al., 2024a) introduced Multi-Track MusicLDM, a latent diffusion model for generating multi-track music. Recently, the authors extended this model to support both multi-track music generation and separation (Karchkhadze et al., 2024b).

7.3 Method

This section details MusicGen-Stem, covering the compression model, the auto-regressive sequential model, and the editing method. Fig. 7.2 describes the training pipeline.

7.3.1 Compression models

For each stem (*drums, bass, other*), we train a compression model similar to EnCodec (Défossez et al., 2023) that compresses 32kHz mono music into tokens at a rate

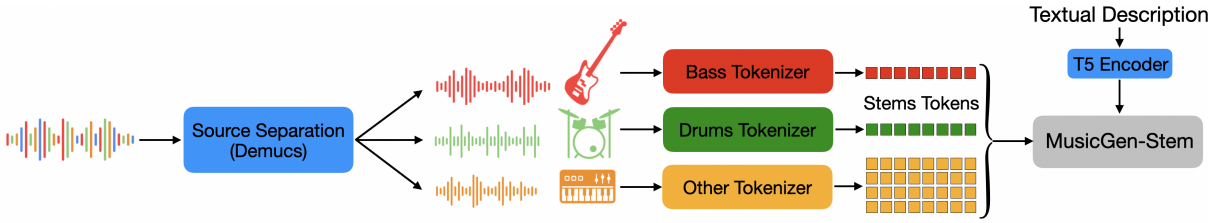

Figure 7.2 – Training pipeline. Given a song paired with its textual description, we process the song by using the source separation model Demucs and tokenize each stem with specific compression models. There is one stream of token for the *bass* as well as the *drums* and 4 streams of tokens for the *other* instruments. Then, these tokens as well as the encoded textual description are fed into MusicGen-Stem's autoregressive transformer which is trained with a cross-entropy loss.

of 50Hz. For the *drums* and the *bass* stem, we obtain good reconstruction quality with only a single quantization level. For the *other* stem, we trained a model with 4 RVQ streams. Each of these specialized models is trained on our internal source separation dataset which consists of 3,000 professionally recorded songs.

7.3.2 Modeling and data preparation

We train an autoregressive transformer for the task of text-to-music generation using cross entropy loss on 30 seconds audio segments at 50Hz. The 3 different stems are tokenized thanks to the 3 different compression models and their streams are concatenated and modeled in parallel. We use the medium (1.5B) architecture of MusicGen's transformer (Copet et al., 2024) and apply its delay pattern to the tokens. In our setup *bass, drums* and the first RVQ *other* stream are "coarse" tokens and in sync. Thus we only apply a delay on the 3 residual streams of the *other* stem (see Fig. 7.3). Given the fact that we do not have a big dataset of labeled stem music, we train our model on the same data as in (Copet et al., 2024) but we removed the songs that had vocals (15% of our data) and use the last version of Demucs (Défossez et al., 2021) to separate all the songs into 3 stems (*bass, drums* and *other*).

7.3.3 Editing

At each training step we either train our model to perform text-to-music generation or editing with a 0.5 probability. To train the editing task, we take a sequence of 25 seconds (1,250 tokens), downsample it by a factor 5 (i.e. 10Hz) and use these 250 tokens as a

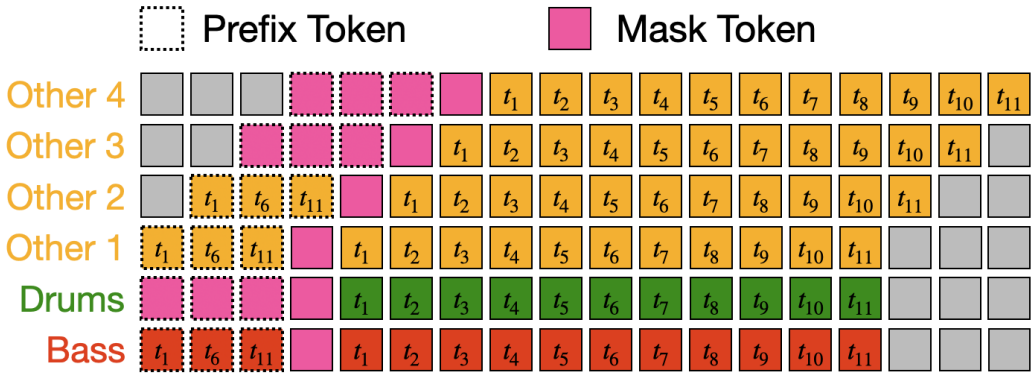

Figure 7.3 – Training the editing task. Here the *drums* and the 2 last streams of the *other* stem are masked. The cross-entropy loss is computed on the tokens on the right of the masked tokens.

prefix for the model. Then, we randomly sample 1 or 2 stems and mask the associated tokens in the prefix. If the *other* stem is selected to be masked, we randomly choose to mask the streams in $\{4\}, \{4, 3\}, \{4, 3, 2\}$, or $\{4, 3, 2, 1\}$, forcing the model to learn to generate the details (the streams 2, 3, 4) of the *other* stem given its first streams. We can see an example of a prefix on Fig. 7.3 where the *drums* is masked as well as the streams 3 and 4 of the *other* stem. At inference time, we can 1) edit a song generated by the model (it is then already tokenized) 2) take an existing song separate its stems with Demucs and tokenize them 3) tokenize single stem music to be able to generate new stems. Then, we downsample to 10Hz this tokens sequence, we mask the desired stem and ask the model to continuous the generation in an autoregressive manner. During the autoregressive generation, we have the choice to force the unmasked streams to be exactly the same and only generate the masked streams or to let the model generate all of the streams. In the second case, we obtain a variation of the original unmasked stems (the model uses the downsampled prefix to reconstruct the stem). As well, the textual prompt let us control the generated new stems.

7.4 Experiments

7.4.1 Training details

MusicGen-Stem is trained for 400K steps with a batch size of 128. The data used for training include the internal MMI dataset that contains 10k high quality songs, Shutter-

Stock and Pond5. We filtered the dataset to remove all songs containing vocals, resulting in a total of 17K hours of instrumental only data. We use AdamW optimizer with a learning rate of 1e-4.

7.4.2 Metrics

We evaluate the proposed method against state-of-the-art music generation models, considering both music generation and stem-editing setups. All the objective evaluations are performed on an internal test set of 534 songs for which we used Demucs to separate the stems.

Music generation: We use established objective and subjective metrics from the literature. Specifically the objective metrics used are: Frechet Audio Distance (Kilgour et al., 2019), the KL-divergence based metric introduced by Copet et al. (2024) and the CLAP score (Elizalde et al., 2023) for text-to-music. For FAD we use the official TensorFlow implementation and pre-trained VGGish model. For subjective evaluations we follow the protocol proposed in MusicGen (Copet et al., 2024) this consists in two studies, one for the overall quality of the samples (OVL) and one for the relation to text (REL).

Music editing: To evaluate the editing performances of our model we perform two objective evaluations. The first one evaluates whether the rhythm matches between the original song and the new stem (BEAT). To do so, we use the beat tracking algorithm from `madmom` (Böck et al., 2016) both on the original song and on the generated stem. Then, we report the F-measure calculated with `mir_eval` (Raffel et al., 2014) using as reference the beats from the original song. To evaluate the harmonic matching (HAR) between the *bass* stem and the *other* stem we use Chordino[1] to extract the chords played in the *other* stem and use Pesto (Riou et al., 2023) to estimate the notes from the *bass* line, we only keep the pitches predicted with a confidence greater than 0.75. We then compute the ratio of time steps where the *bass* is playing a chord tone note (Persichetti, 1961). For both metrics we zero out stems when the loudness was lower than -35dB.

In addition, we also conducted three subjective assessments of the editing process. In each assessment, we replaced one of the stems of the original song with a generated one. To ensure raters clearly ear the difference, we boost the generated stem to match the loudness of the rest of the track. We then set the overall loudness of this mix to -14dB. Participants are told which instrument differs and are asked to rate the overall quality of

Model	FAD ↓	CLAP ↑	KLD ↓	REL ↑	OVL ↑
Ground Truth	×	0.40	×	93.4 ±0.7	93.6 ±0.5
MusicGen*	0.75	0.37	**0.59**	84.4 ±1.0	86.7 ±0.8
MusicGen-Stem*	**0.70**	**0.38**	0.60	**85.4** ±0.7	**87.0** ±0.8
MusicGen	2.13	×	1.02	×	**85.0** ±0.7
MSDM RT	14.05	×	1.19	×	84.7 ±0.8
MSDM PT	7.61	×	1.48	×	80.9 ±1.0
MusicGen-Stem	2.15	×	1.04	×	83.8 ±0.9

Table 7.1 – Comparisons of the different music generation models first in a text conditioned setup and then in an unconditional setup. Use of text conditioning is indicated with *.

the resulting songs. Every subjective study includes 40 samples that are each rated by at least 4 participants.

7.4.3 Text-conditioned and unconditional music generation

In this section, we benchmark two families of generative models for music: text conditioned models and stem-level models. Note that only MusicGen-Stem fits into both categories since MSDM does not handle text conditioning and MusicGen operates at the mixture level. To evaluate MSDM (Mariani et al., 2024) we use the official implementation[2]. Since the original model (PT) is trained on the limited Slakh2100 dataset (Manilow et al., 2019), we include a version of this model trained on our dataset (RT).

Objective and subjective metrics presented in TABLE 7.1 suggest that MusicGen-Stem is on par with its predecessor on text-conditioned music generation. In the unconditional setup, our results suggest that MusicGen and MusicGen-Stem perform on par. OVL scores shows that MSDM RT produces good quality outputs. However, this model primarily produces similar songs, mainly ambient tracks with subdued *drums* and *bass*. This lack of diversity is evident in its FAD score exceeding 14.

7.4.4 Text-conditioned and unconditional music editing

We evaluate MusicGen-Stem on single stem music editing. The model is used to generate a coherent third stem in the context of two given stems. We compare it to both versions of MSDM and Instruct-MusicGen. Since the latter regenerates everything at the

1. https://github.com/ohollo/chord-extractor
2. https://github.com/gladia-research-group/multi-source-diffusion-models

Edited stem	HAR ↑ bass	HAR ↑ other	BEAT ↑ bass	BEAT ↑ drums	BEAT ↑ other	OVL ↑ bass	OVL ↑ drums	OVL ↑ other
Ground Truth	72%	72%	0.52	0.87	0.55	93.9 ±0.7	93.4 ±0.7	93.5 ±0.6
MSDM RT	48%	47%	0.28	0.18	**0.45**	72.9 ±1.6	54.0 ±2.0	54.7 ±1.6
MSDM PT	31%	41%	0.03	0.20	0.04	65.0 ±2.0	54.6 ±2.4	42.4 ±2.7
Instruct-MusicGen	N/A	N/A	N/A	N/A	N/A	83.9 ±0.9	51.4 ±1.1	64.4 ±1.1
MusicGen-Stem*	**66%**	**68%**	0.42	**0.69**	0.41	86.5 ±0.8	**86.8** ±0.9	**75.8** ±1.7
MusicGen-Stem	**66%**	67%	**0.46**	0.67	**0.45**	**86.7**±0.9	86.4 ±0.8	72.6 ±1.2

Table 7.2 – Performances of the models on stem editing task. Use of text conditioning is indicated with *.

mixture level, it does not keep the input stems unchanged which prevents us to compute objective metrics.

Results from Table 7.2 indicate that MusicGen-Stem outperforms all evaluated baselines in stem editing, regardless of whether text conditioning is applied. Our model consistently generates stems that are more coherent with the overall track, both in terms of rhythm and pitch. Subjective evaluations further validate the superior editing performance of our model. While Instruct-MusicGen shows promising results in bass performance, it is constrained to 5-second audio clips and occasionally alters the song significantly.

7.5 Conclusion

We introduce a model that is capable of generating music conditioned on either text or instrument stems. MusicGen-Stem reaches comparable performance to the evaluated baselines when considering text-to-music generation, while allowing stem editing. This makes it possible for musicians to iterate on their creations by being able to keep some parts at the instrument level. While MusicGen-Stem is a step towards better control in music generation, it is still limited to 3 stems due to the lack of high quality dataset containing more than the classic *bass*, *drums* and *other*. For future work we intend to increase the capacity of the *bass* tokenizer that tends to create artefacts for higher pitch notes. We also want to have better control on the *other* stem generation with refined conditioning like instrument embedding.

Part II

Watermarking for Safer Audio Generative Modeling

Chapter 8

Introduction to Audio Watermarking

Watermarking is a technique that has been used for a long time in the field of digital forensics. It consists in modifying a content in a way that can be retrieved by the author of the modification. Traditionally this has been used to attest ownership of the content or to hide secret messages into data. For example, in audio, the video game *Doom* famously hid a satanic symbol in the spectrogram of the audio (Figure 8.1).

In the context of the rapidly evolving generative AI technologies, models capable of generating photorealistic images (Ho et al., 2020; Saharia et al., 2022), producing text at a quality undistinguishable from humans (Brown et al., 2020b; Touvron et al., 2023) or synthesising natural sounding human voices (Le et al., 2024; Vyas et al., 2023; Wang et al., 2023a) have emerged. These new capabilities of generative models, specifically in the generative AI field, have brought a new set of risks for society. While very recent, these technologies have already been used by malicious actors aiming to influence opinion (Goldstein et al., 2023), commit scam, and spread misinformation (Spitale et al., 2023).

Societal issues related to the use of generative AI led to a growing interest in AI security and responsible AI. One critically important field of interest for this field of research is the ability to distinguish fake, generated content from real one. This is particularly important in the context of generative AI, where the generated content can be very realistic and difficult to distinguish from real content.

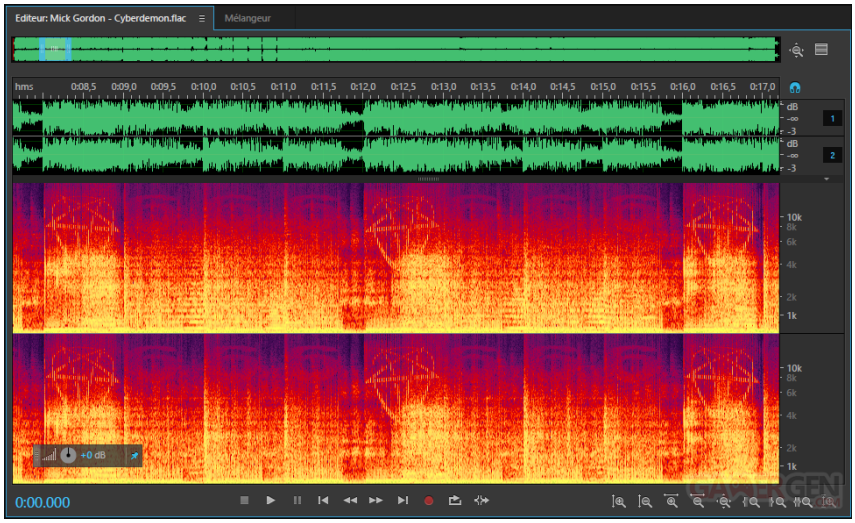

Figure 8.1 – The spectrogram of the audio of the video game *Doom* contains a hidden satanic image (Statt, 2016).

In forensics, the task of simply classifying real vs. fake content is called *passive detection* (Rössler et al., 2019). This task is very attractive because it has the advantage of not requiring any additional effort from the generating side, all the effort being placed at the detector level. This task is known to be very difficult, detectors tending to overfit on certain generator architectures or specific models. The task is expected to get harder with the improvement and multiplication of generative AI models.

One of the traditional forensic area of research is Watermarking. The idea of watermarking is to modify content to hide information in it. While it may seem unrelated to the deep fake detection task it can be used in this purpose. This can be done by using the watermark to hide as a secret message certifying the data comes from an AI model.

Using watermarking for the detection of fake content is called *active detection* because it relies on a proactive mechanism consisting in modifying the generated content to ease its detection down the line. This strategy makes it much easier for the detector but has its own set of challenges that will be listed in the next section.

8.1 Challenges

As described in the last section, watermarking can be used as a way to enhance detection performances from generative models. While watermarking can be used in many

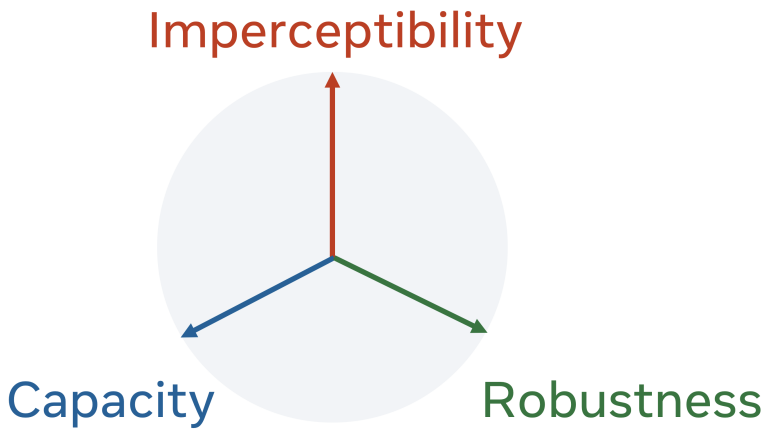

Figure 8.2 – The three criteria of watermarking

different contexts, this manuscript focuses specifically on use cases related to generative AI. This section goes over the three orthogonal criteria for good watermarking method as shown on Figure 8.2. The three criteria are **capacity**, **robustness** and **imperceptibility**.

8.1.1 Capacity

The first and main criteria is called the capacity of the watermark, this relates to the size of the message that can be hidden inside a particular data content. This is often measured in term of number of bits. In the context of generative AI the capacity of the watermark can be used for instance to identify users, or versions of a model. The capacity is also related to the confidence of the detection using statistical tests. If we suppose that messages are equally distributed among natural images, which is a standard assumption in the watermarking literature, then the probability of a natural image having exactly the watermarking message which is refered to as false positive is :

$$P = \left(\frac{1}{2}\right)^N \tag{8.1}$$

where N is the capacity of the watermark expressed in number of bits. This is also known as the theoretical false positive rate for a watermarking model. However, it is not always the case that the messages are really evenly distributed in natural data which can result in a gap between theoretical and empirical false positive rate. It can also result in a false positive rate that will depend on the chosen message.

8.1.2 Imperceptibility

The second way of evaluating the quality of a watermarking method is to measure perceptually the modification caused by the watermark. Simply put, the applied watermark should not degrade too much the performances of the generative models. However, measuring audible differences is not an easy task that can sometimes even be dependent on individual attributes of the listeners (sensitivity to frequency ranges, etc..). Both objective and subjective evaluation can be performed, as introduced in Section 4.3.

8.1.3 Robustness

Robustness is the last important factor when it comes to the quality of a watermarking method. The robustness of the watermarking is measured by how the performance of watermarking detection are affected when the audio content is altered. We can distinguish two types of modifications: standards modifications unrelated to the watermark that we call *edits*, and modifications performed by a malicious user aiming at removing or forging the watermark that we call *attacks*.

Robustness to edits

Robustness to edits is a basic requirement of watermarking methods, audio edits are the typical modifications of a signal that can be performed by regular users. Typical examples include the introduction of white noise or reverberation when the audio is re-recorded with a device, the use of audio compression (see Section 1.3) or the change of sampling rate. Those modifications can even be performed by softwares or social networks without the user being aware of it happening.

To implement watermarking at scale *e.g.* on social media platforms, it is often considered a baseline requirement to be very robust to audio edits. This will cover the vast majority of the modifications that will be performed on the audio content by regular users.

Robustness to attacks

As described before, attacks are modifications performed by malicious users, specifically to remove or to introduce the watermark into an audio. This task is related to the literature on adversarial attacks (Wang et al., 2023b). It is often considered impossible to be robust to all attacks, since attackers can always come up with new ways to remove

the watermark.

8.1.4 Conflicting objectives

The three criteria described above are often conflicting, for instance increasing the capacity of the watermarking will often result in a decrease of the imperceptibility, and the message detection will be more sensitive to edits making the watermark less robust. Similarly, making the watermark more imperceptible will often be by reducing the magnitude of the watermark distortion, which may reduce the possibility of high message capacity or strong robustness. This is why it is important to find a good trade-off between the three criteria.

8.2 Watermarking for Generative AI content

In this manuscript, we focus on watermarking methods for generative AI content, specifically audio. This section goes over specificities of watermarking for generative AI content.

Importance of the encoded message. While it can be useful to hide a secret message in the content (e.g. to link the content to a specific user or model), in the context of generative AI, the main goal is often to detect whether the content was generated by an AI model or not. In this case, the capacity of the model is not the main concern. Models that only focus on detection can reduce the message capacity to zero. This type of watermarking is referred to as *0-bit watermarking*.

Watermarking at different stages of the generation process. In the context of generative AI, watermarking can be performed at different stages of the generation process. It can be applied to the output of the generative model, called *post-hoc*, or one can make sure that the model natively outputs watermarked content. This is referred to as *in-model* watermarking. *Post-hoc* watermarking is easier to implement since it does not require any modification of the generative model.

However, since the generation and watermarking stage are performed separately, it becomes very easy for an attacker to remove the watermark by simply skipping the watermarking stage. This major drawback makes *post-hoc* watermarking only suitable for

closed models where the generation and watermarking stage are performed by the same entity behind an API. In-model watermarking tends to be more difficult to implement since it requires modifying or retraining the generative model. With this strategy, generation and watermarking stages are entangled and thus it is not easy to perform generation without watermarking making it suitable in more diverse situations even when the model is open source.

In this chapter, we go over the current literature on watermarking methods for audio and for images, since the two domains share many similarities and some advancements in audio watermarking come from the image literature. We will first cover classical watermarking methods, then deep-learning-based watermarking methods.

8.3 Traditional Watermarking Methods

Originally, watermarking methods were developed using ideas from signal processing and information theory. These methods can typically be categorized into two main types: spatial domain watermarking and frequency domain watermarking. Spatial-domain watermarking embeds the watermark directly into the pixel values of an image (Ni et al., 2006; Nikolaidis and Pitas, 1998; Van Schyndel et al., 1994) or the samples of an audio signal (Lie and Chang, 2006). This is typically done by modifying the least significant bits (LSBs) of the pixel values or audio samples, which makes the watermark less perceptible to human observers.

The other alternative is to apply the watermark in some transformed domain, such as the frequency domain, using the Discrete Fourier Transform, Discrete Cosine Transform (DCT) or the Discrete Wavelet Transform (DWT). The standard approach is called the spread spectrum technique (Barni and Bartolini, 2004; Cox et al., 1997), which spreads the watermark signal across a wide frequency range by adding a pseudo-random noise sequence. At detection time, the receiver can extract the watermark by correlating the received signal with the pseudo random sequence. In audio, frequency domain watermarking is often preferred because it can be more robust to common audio processing operations and tends to be less sensitive to high frequency noise.

When trying to detect watermarks in the audio, one challenge is often to retrieve the original alignment of the watermark with the audio signal i.e. the start of the segment

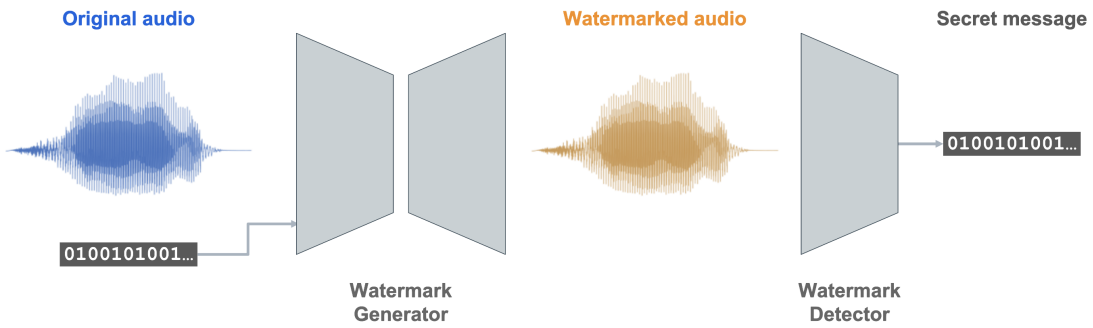

Figure 8.3 – Deep Learning Watermarking Method. Simplified illustration of an HIDDEN-like (Zhu et al., 2018) generator-detector watermarking method for audio.

that was watermarked This problem is often referred to as the synchronization problem. One solution to this problem is to use repeated synchronization bits (Xiang et al., 2014), at decoding time, the receiver can make sure that the watermark is correctly aligned with the audio signal by checking that those synchronization bits are indeed repeated. This allows to detect the location of potential attacks on the watermark, such as time shifting or cropping. Another approach is to use block repetitions (Kirovski and Attias, 2002; Kirovski and Malvar, 2003). This method is similar to the previous one, but since blocks are repeated in the time-frequency domain, it is also robust to frequency-shifting attacks.

8.4 Deep Learning Watermarking Methods

Using deep learning for watermarking has become increasingly popular in recent years, especially with the rise of generative models. The most common approach is to perform *post-hoc* watermarking by training two models: a watermark embedder and a watermark detector. In this setup, the embedder takes an input signal (e.g., an audio or an image) and binary message and produces a watermarked signal that embeds the message. The detector then takes the watermarked signal and tries to recover the original message. The two models are trained jointly in an end-to-end fashion, by minimizing the distortion while maximizing message recovery. This end-to-end approach allows one to train for the robustness to any transformation by just performing the transformation at training time between the embedder and the detector. Hidden (Zhu et al., 2018) is one of the first works to propose this approach in the image domain, it uses a convolutional neural network (CNN) to embed the watermark in images and a second CNN to detect it. In the audio domain many works have built on this idea (O'Reilly et al., 2024; Pavlović et al., 2022).

Wavmark (Chen et al., 2024a) notably proposes a way to ensure minimal SNR performance by keeping the watermark only on windows where the degradation is lower than a given threshold. Additionally, the authors introduce an every-epoch rebalancing of the training augmentation sampling ratio to emphasize augmentations where the model performed poorly during validation. DEAR (Liu et al., 2023a) specifically focuses on robustness to re-recording, which is hard to simulate in training. Timbre Watermarking (Liu et al., 2023b), proposes a speech-focused method aimed at preventing voice cloning attacks by using complex distortion layers during training. Liu et al. (2024c) propose a method that perturbs the initial noise of a diffusion-based vocoder conditioned on the mel spectrogram of the original audio.

In-model watermarking To allow open sourcing of generative models, some works have proposed that those models embed a watermark directly at generation time. Early approaches proposed to watermark the training dataset (Wu et al., 2020; Yu et al., 2021), we adapt this idea to audio language models in Chapter 10. Another solution is to train the generative model with a watermarking loss (Fei et al., 2022; Juvela and Wang, 2024b), in the context of latent generative models, this approach can be applied to the decoder of such models (Fernandez et al., 2023).

Other generation-time approaches Some approaches are neither post-hoc nor in-model, but rather apply the watermarking by altering the generative process of non-watermarked models. This techniques are notably used to watermark outputs of large language models (Kirchenbauer et al., 2023), the idea is to modify the sampling distribution of the language model. In the case of diffusion models, this can be achieved by modifying the initial noise distribution(Wen et al., 2023) or crafting intermediate states with adversarial optimization (Huang et al., 2024).

Chapter 9

Improving Audio Watermarking Models (Audioseal)

This chapter describes the paper "Proactive Detection of Voice Cloning with Localized Watermarking" (San Roman et al., 2024) published at ICML 2024. It introduces a novel post-hoc audio watermarking method, AudioSeal, that enables localized and robust detection of AI-generated audio.

9.1 Introduction

Generative speech models are now capable of synthesizing voices that are indistinguishable from real ones (Arik et al., 2018; Casanova et al., 2023; Kim et al., 2021; Wang et al., 2023a). Though speech generation and voice cloning are not novel concepts, their recent advancements in quality and accessibility have raised new security concerns. A notable incident occurred where a deepfake audio misleadingly urged US voters to abstain, showcasing the potential for misusing these technologies to spread false information (Murphy et al., 2024). Regulators and governments are implementing measures for AI content transparency and traceability, including forensics and watermarking – see Commission (2023); of China (2023); USA (2023).

The main forensics approach to detect synthesized audio is to train binary classifiers to

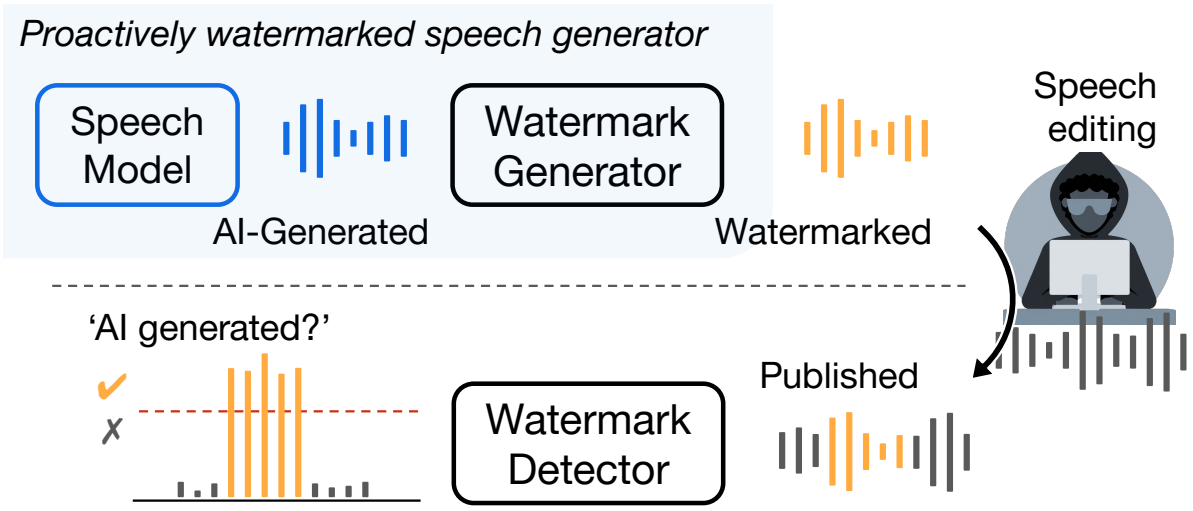

Figure 9.1 – Proactive detection of AI-generated speech. We embed an imperceptible watermark in the audio, which can be used to detect if a speech is AI-generated and identify the model that generated it. It can also precisely pinpoint AI-generated segments in a longer audio with a sample level resolution (1/16k seconds).

discriminate between natural and synthesized audios, a technique highlighted in studies by Borsos et al. (2022); Kharitonov et al. (2023b); Le et al. (2024). We refer to this technique as *passive detection* since it does not alter of the audio source. Albeit being a straightforward mitigation, it is prone to fail as generative models advance and the difference between synthesized and authentic content diminishes.

Watermarking emerges as a strong alternative. It embeds a signal in the generated audio, imperceptible to the ear but robustly detectable by specific algorithms. There are two watermarking types: multi-bit and zero-bit. Zero-bit watermarking detects the presence or absence of a watermarking signal, which is valuable for AI content detection. Multi-bit watermarking embeds a binary message in the content, allowing to link content to a specific user or generative model. Most deep-learning based audio watermarking methods (Chen et al., 2024a; Liu et al., 2023a; Pavlović et al., 2022) are multi-bit. They train a generator to output the watermarked audio from a sample and a message, and an extractor retrieving the hidden message.

Current watermarking methods have limitations. First, *they are not adapted for detection.* The initial applications assumed any sound sample under scrutiny was watermarked (*e.g.* IP protection). As a result, the decoders were never trained on non-watermarked samples. This discrepancy between the training of the models and their practical use

leads to poor or overestimated detection rates, depending on the embedded message (see App. 9.7). Our method aligns more closely with the concurrent work by Juvela and Wang (2024a), which trains a detector, rather than a decoder.

Second, they *are not localized* and consider the entire audio, making it difficult to identify small segments of AI-generated speech within longer audio clips. The concurrent WavMark's approach (Chen et al., 2024a) addresses this by repeating at 1-second intervals a synchronization pattern followed by the actual binary payload. This has several drawbacks. It cannot be used on spans less than 1 second and is susceptible to temporal edits. The synchronization bits also reduce the capacity for the encoded message, accounting for 31% of the total capacity. Most importantly, the brute force detection algorithm for decoding the synchronization bits is prohibitively slow especially on non-watermarked content, as we show in Sec. 9.4.5. This makes it unsuitable for real-time and large-scale traceability of AI-generated content on social media platforms, where most content is not watermarked.

To address these limitations, we introduce *AudioSeal*, a method for localized speech watermarking. It jointly trains two networks: a *generator* that predicts an additive watermark waveform from an audio input, and a *detector* that outputs the probability of the presence of a watermark at each sample of the input audio. The detector is trained to precisely and robustly detect synthesized speech embedded in longer audio clips by masking the watermark in random sections of the signal. The training objective is to maximize the detector's accuracy while minimizing the perceptual difference between the original and watermarked audio. We also extend AudioSeal to multi-bit watermarking, so that an audio can be attributed to a specific model or version without affecting the detection signal.

We evaluate the performance of AudioSeal to detect and localize AI-generated speech. AudioSeal achieves state-of-the-art results on robustness of the detection, far surpassing passive detection with near perfect detection rates over a wide range of audio edits. It also performs sample-level detection (at resolution of 1/16k second), outperforming WavMark in both speed and performance. In terms of efficiency, our detector is run once and yields detection logits at every time-step, allowing for real-time detection of watermarks in audio streams. This represents a major improvement compared to earlier watermarking methods, which require synchronizing the watermark within the detector,

thereby substantially increasing computation time. Finally, in conjunction with binary messages, AudioSeal almost perfectly attributes an audio to one model among $1,000$, even in the presence of audio edits.

Our overall contributions are:

— We introduce AudioSeal, the first audio watermarking technique designed for localized detection of AI-generated speech up to the sample-level;

— A novel perceptual loss inspired by auditory masking, that enables AudioSeal to achieve better imperceptibility of the watermark signal;

— AudioSeal achieves the state-of-the-art robustness to a wide range of real life audio manipulations (section 9.4);

— AudioSeal significantly outperforms the state-of-the-art models in computation speed, achieving up to two orders of magnitude faster detection (section 9.4.5);

— Insights on the security and integrity of audio watermarking techniques when open-sourcing (section 9.5).

9.2 Method

The method jointly trains two models. The generator creates a watermark signal that is added to the input audio. The detector outputs local detection logits. The training optimizes two concurrent classes of objectives: minimizing the perceptual distortion between original and watermarked audios and maximizing the watermark detection. To improve robustness to modifications of the signal and localization, we include a collection of train time augmentations. At inference time, the logits precisely localize watermarked segments allowing for detection of AI-generated content. Optionally, short binary identifiers may be added on top of the detection to attribute a watermarked audio to a version of the model while keeping a single detector.

9.2.1 Training pipeline

Figure 9.2 illustrates the joint training of the generator and the detector with four critical stages:

1. The watermark generator takes as input a waveform $s \in \mathbb{R}^T$ and outputs a watermark waveform $\delta \in \mathbb{R}^T$ of the same dimensionality, where T is the number of

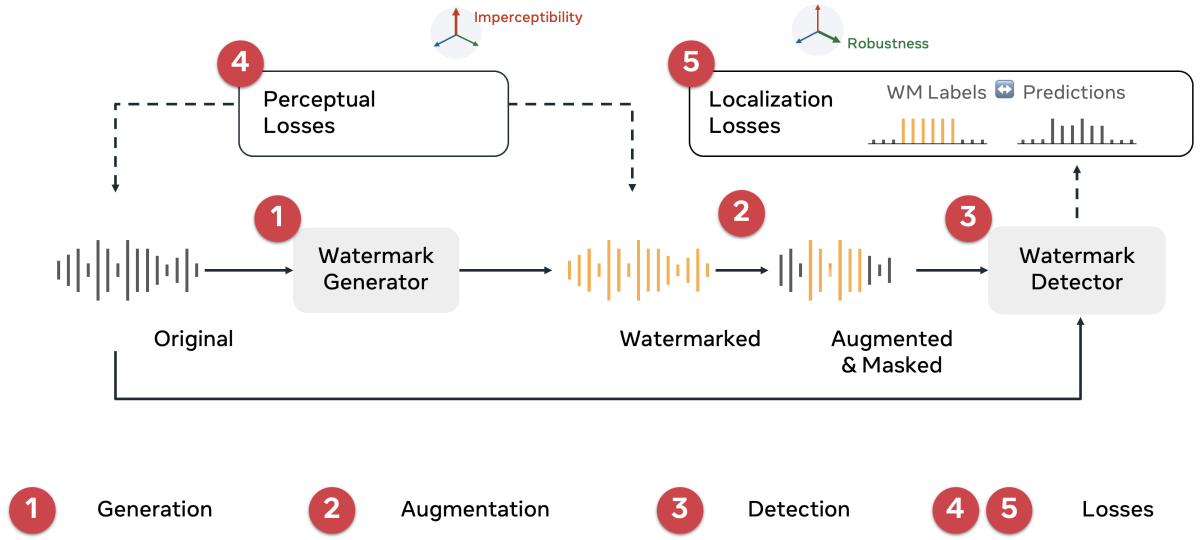

Figure 9.2 – Generator-detector training pipeline.

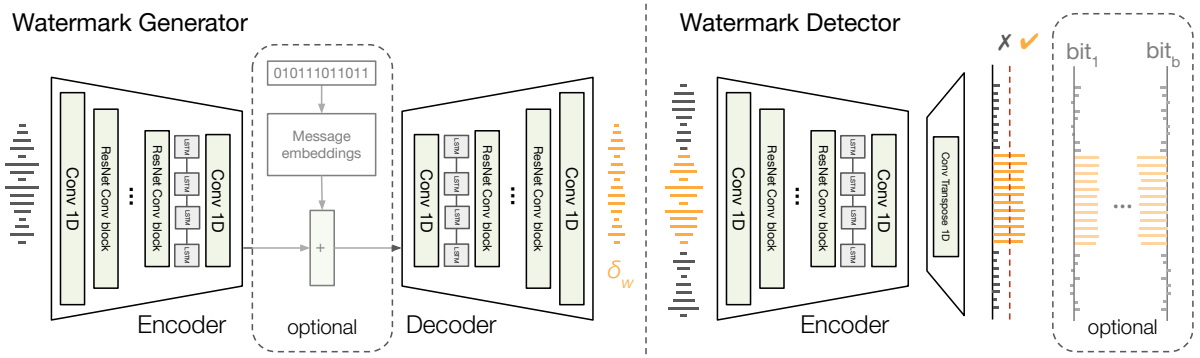

Figure 9.3 – **Architectures**. The *generator* is made of an encoder and a decoder both derived from EnCodec's design, with optional message embeddings. The encoder includes convolutional blocks and an LSTM, while the decoder mirrors this structure with transposed convolutions. The *detector* is made of an encoder and a transpose convolution, followed by a linear layer that calculates sample-wise logits. Optionally, multiple linear layers can be used for calculating k-bit messages. More details in App. 9.9.3.

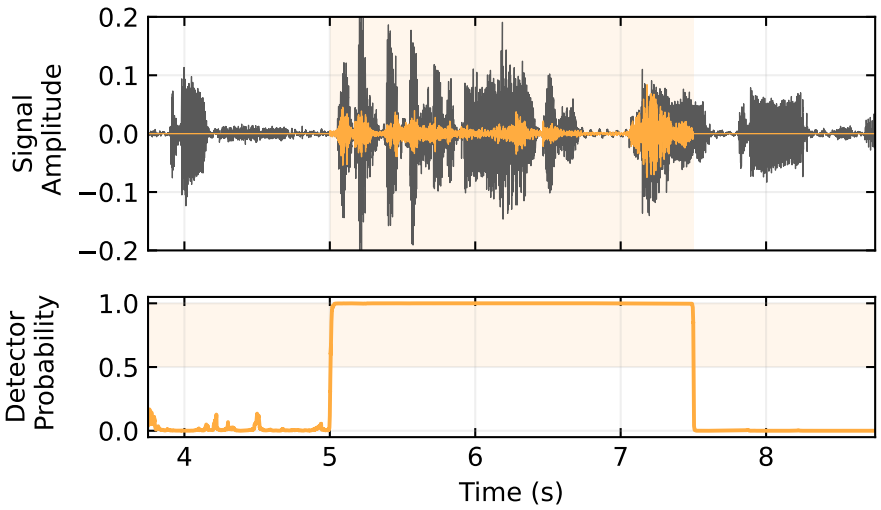

Figure 9.4 – (Top) A speech signal (gray) where the watermark is present between 5 and 7.5 seconds (orange, magnified by 5). (Bottom) The output of the detector for every time step. An orange background color indicates the presence of the watermark.

samples in the signal. The watermarked audio is then $s_w = s + \delta$.

2. To enable sample-level localization, we adopt an augmentation strategy focused on watermark masking with silences and other original audios. This is achieved by randomly selecting k starting points and altering the next $T/2k$ samples from s_w in one of 4 ways: revert to the original audio (*i.e.* $s_w(t) = s(t)$) with probability 0.4; replacing with zeros (*i.e.* $s_w(t) = 0$) with probability 0.2; or substituting with a different audio signal from the same batch (*i.e.* $s_w(t) = s'(t)$) with probability 0.2, or not modifying the sample at all with probability 0.2.
3. The second class of augmentation ensures the robustness against audio editing. One of the following signal alterations is applied: bandpass filter, boost audio, duck audio, echo, highpass filter, lowpass filter, pink noise, gaussian noise, slower, smooth, resample (full details in App. 9.9.2). The parameters of those augmentations are fixed to aggressive values to enforce maximal robustness and the probability of sampling a given augmentation is proportional to the inverse of its evaluation detection accuracy. We implemented these augmentations in a differentiable way when possible, and otherwise (*e.g.* MP3 compression) with the straight-through estimator (Yin et al., 2019) that allows the gradients to back-propagate to the generator.
4. Detector D processes the original and the watermarked signals, outputting for each a soft decision at every time step, meaning $D(s) \in [0, 1]^T$. Figure 9.4 illustrates

that the detector's outputs are at one only when the watermark is present.

The architectures of the models are based on EnCodec (Défossez et al., 2023). They are presented in Figure 9.3 and detailed in the appendix 9.9.3.

9.2.2 Losses

Our setup includes multiple perceptual losses and a localization loss. We balance them during training by scaling their gradients as done by Défossez et al. (2023). The complete list of used losses is detailed below.

Perceptual losses enforce the watermark imperceptibility to the human ear. These include an ℓ_1 loss on the watermark signal to decrease its intensity, the multi-scale Mel spectrogram loss of (Gritsenko et al., 2020), and discriminative losses based on adversarial networks that operate on multi-scale short-term-Fourier-transform spectrograms. Défossez et al. (2023) use this combination of losses for training the EnCodec model for audio compression.

In addition, we introduce a novel time-frequency loudness loss **TF-Loudness**, which operates entirely in the waveform domain. This approach is based on "auditory masking", a psycho-acoustic property of the human auditory system already exploited in the early days of watermarking (Kirovski and Attias, 2003): the human auditory system fails perceiving sounds occurring at the same time and at the same frequency range (Schnupp et al., 2011). TF-Loudness is calculated as follows: first, the input signal s is divided into B signals based on non-overlapping frequency bands $s_0, \ldots, s_{B-1}$. Subsequently, every signal is segmented using a window of size W, with an overlap amount denoted by r. This procedure is applied to both the original audio signal s and the embedded watermark δ. As a result, we obtain segments of the signal and watermark in time-frequency dimensions, denoted as s_b^w and δ_b^w respectively. For every time-frequency window we compute the loudness difference, where loudness is estimated using ITU-R BS.1770-4 recommendations (telecommunication Union, 2011) (see App. 9.9.1 for details):

$$l_b^w = \text{Loudness}(\delta_b^w) - \text{Loudness}(s_b^w). \tag{9.1}$$

This measure quantifies the discrepancy in loudness between the watermark and the original signal within a specific time window w, and a particular frequency band b. The

final loss is a weighted sum of the loudness differences using softmax function:

$$\mathcal{L}_{loud} = \sum_{b,w} \left(\text{softmax}(l)_b^w * l_b^w\right). \tag{9.2}$$

The softmax prevents the model from targeting excessively low loudness where the watermark is already inaudible.

Masked sample-level detection loss. A localization loss ensures that the detection of watermarked audio is done at the level of individual samples. For each time step t, we compute the binary cross entropy (BCE) between the detector's output $D(s)_t$ and the ground truth label (0 for non-watermarked, 1 for watermarked). Overall, this reads:

$$\mathcal{L}_{loc} = \frac{1}{T}\sum_{t=1}^{T} \text{BCE}(D(s')_t, y_t), \tag{9.3}$$

where s' might be s or s_w, and where time step labels y_t are set to 1 if they are watermarked, and 0 otherwise.

9.2.3 Multi-bit watermarking

We extend the method to support multi-bit watermarking, which allows for attribution of audio to a specific model version. *At generation*, we add a message processing layer in the middle of the generator. It takes the activation map in $\mathbb{R}^{h,t'}$ and a binary message $m \in \{0,1\}^b$ and outputs a new activation map to be added to the original one. We embed m into $e = \sum_{i=0..b-1} E_{2i+m_i} \in \mathbb{R}^h$, where $E \in \mathbb{R}^{2b,h}$ is a learnable embedding layer. e is then repeated t times along the temporal axis to match the activation map size (t, h). *At detection*, we add b linear layers at the very end of the detector. Each of them outputs a soft value for each bit of the message at the sample-level. Therefore, the detector outputs a tensor of shape $\mathbb{R}^{t,1+b}$ (1 for the detection, b for the message). *At training*, we add a decoding loss $\mathcal{L}_{dec}$ to the localization loss $\mathcal{L}_{loc}$. This loss $\mathcal{L}_{dec}$ averages the BCE between the original message and the detector's outputs over all parts where the watermark is present.

9.2.4 Training details

Our watermark generator and detector are trained on a 4.5K hours subset from the VoxPopuli (Wang et al., 2021) dataset. It is important to emphasize that the sole purpose of our generator is to generate imperceptible watermarks given an input audio; without the capability to produce or modify speech content. We use a sampling rate of 16 kHz and one-second samples, so $T = 16000$ in our training. A full training requires 600k steps, with Adam, a learning rate of 10^{-4}, and a batch size of 32. For the drop augmentation, we use $k = 5$ windows of 0.1 sec. h is set to 32, and the number of additional bits b to 16 (note that h needs to be higher than b, for example $h = 8$ is enough in the zero-bit case). The perceptual losses are balanced and weighted as follows: $\lambda_{\ell_1} = 0.1$, $\lambda_{msspec} = 2.0$, $\lambda_{adv} = 4.0$, $\lambda_{loud} = 10.0$. The localization and watermarking losses are weighted by $\lambda_{loc} = 10.0$ and $\lambda_{dec} = 1.0$ respectively.

9.2.5 Detection, localization and attribution

At inference, we may use the generator and detector for:

— *Detection*: To determine if the audio is watermarked or not. To achieve this, we use the average detector's output over the entire audio and flag it if the score exceeds a threshold (default: 0.5).

— *Localization*: To precisely identify where the watermark is present. We utilize the sample-wise detector's output and mark a time step as watermarked if the score surpasses a threshold (default: 0.5).

— *Attribution*: To identify the model version that produced the audio, enabling differentiation between users or APIs with a single detector. The detector's first output gives the detection score and the remaining k outputs are used for attribution. This is done by computing the average message over detected samples and returning the identifier with the smallest Hamming distance.

9.3 Audio/Speech Quality

We first evaluate the quality of the watermarked audio using: Scale Invariant Signal to Noise Ratio (SI-SNR): $\text{SI-SNR}(s, s_w) = 10 \log_{10} (\|\alpha s\|_2^2 / \|\alpha s - s_w\|_2^2)$, where $\alpha = \langle s, s_w \rangle / \|s\|_2^2$; as well as PESQ (Rix et al., 2001), ViSQOL (Chinen et al., 2020) and

Table 9.1 – Audio quality metrics. Compared to traditional watermarking methods that minimize the SNR like WavMark, AudioSeal achieves same or better perceptual quality.

Methods	SI-SNR	PESQ	STOI	ViSQOL	MUSHRA
WavMark	**38.25**	4.302	0.997	4.730	71.52 ± 7.18
AudioSeal	26.00	**4.470**	0.997	**4.829**	**77.07** ± 6.35

STOI (Taal et al., 2010) which are objective perceptual metrics measuring the quality of speech signals.

Table 9.1 report these metrics. AudioSeal behaves differently than watermarking methods like WavMark (Chen et al., 2024a) that try to minimize the SI-SNR. In practice, high SI-SNR is indeed not necessarily correlated with good perceptual quality. AudioSeal is not optimized for SI-SNR but rather for perceptual quality of speech. This is better captured by the other metrics (PESQ, STOI, ViSQOL), where AudioSeal consistently achieves better performance. Put differently, our goal is to hide as much watermark power as possible while keeping it perceptually indistinguishable from the original. Figure 9.4 also visualizes how the watermark signal follows the shape of the speech waveform.

The metric used for our subjective evaluations is MUSHRA test (Series, 2014a). The complete details about our full protocol can be found in the Appendix 9.9.4. In this study our samples got ratings very close to the ground truth samples that obtained an average score of 80.49.

9.4 Experiments and Evaluation

This section evaluates the detection performance of passive classifiers, watermarking methods, and AudioSeal, using True Positive Rate (TPR) and False Positive Rate (FPR) as key metrics for watermark detection. TPR measures correct identification of watermarked samples, while FPR indicates the rate of genuine audio clips falsely flagged. In practical scenarios, minimizing FPR is crucial. For example, on a platform processing 1 billion samples daily, an FPR of 10^{-3} and a TPR of 0.5 means that 1 million samples require manual review each day, yet only half of the watermarked samples are detected.

Table 9.2 – Comparison with Voicebox binary classifier. Percentage refers to the fraction of masked input frames.

	AudioSeal (Ours)			**Voicebox Classif.**		
% Mask	Acc.	TPR	FPR	Acc.	TPR	FPR
Original audio vs AI-generated audio						
30%	1.0	1.0	0.0	1.0	1.0	0.0
50%	1.0	1.0	0.0	1.0	1.0	0.0
90%	1.0	1.0	0.0	1.0	1.0	0.0
Re-synthesized audio vs AI-generated audio						
30%	**1.0**	**1.0**	**0.0**	0.704	0.680	0.194
50%	**1.0**	**1.0**	**0.0**	0.809	0.831	0.170
90%	**1.0**	**1.0**	**0.0**	0.907	0.942	0.112

9.4.1 Comparison with passive classifier

We first compare detection results on samples generated with Voicebox (Le et al., 2024). We compare to the passive setup where a classifier is trained to discriminate between Voicebox-generated and real audios. Following the approach in the Voicebox study, we evaluate 2,000 approximately 5-second samples from LibriSpeech, These samples have masked frames (90%, 50%, and 30% of the phonemes) pre-Voicebox generation. We evaluate on the same tasks, *i.e.* distinguishing between original and generated, or between original and re-synthesized (created by extracting the Mel spectrogram from original audio and then vocoding it with the HiFi-GAN vocoder).

Both active and passive setups achieve perfect classification in the case when trained to distinguish between natural and Voicebox. Conversely, the second part of Tab. 9.2 highlights a significant drop in performance when the classifier is trained to differentiate between Voicebox and re-synthesized. It suggests that the classifier is detecting vocoder artifacts, since the re-synthesized samples are sometimes wrongly flagged. The classification performance quickly decreases as the quality of the AI-generated sample increases (when the input is less masked). On the other hand, our proactive detection does not rely on model-specific artifacts but on the watermark presence. This allows for perfect detection over all the audio clips.

Table 9.3 – Detection results for different edits applied before detection. Acc. (TPR/FPR) is the accuracy (and TPR/FPR) obtained for the threshold that gives best accuracy on a balanced set of augmented samples. AUC is the area under the ROC curve.

	AudioSeal (Ours)		**WavMark**	
Edit	Acc. TPR/FPR	AUC	Acc. TPR/FPR	AUC
None	1.00 1.00/0.00	1.00	1.00 1.00/0.00	1.00
Bandpass	1.00 1.00/0.00	1.00	1.00 1.00/0.00	1.00
Highpass	0.61 0.82/0.60	0.61	**1.00 1.00/0.00**	**1.00**
Lowpass	**0.99 0.99/0.00**	**0.99**	0.50 1.00/1.00	0.50
Boost	1.00 1.00/0.00	1.00	1.00 1.00/0.00	1.00
Duck	1.00 1.00/0.00	1.00	1.00 1.00/0.00	1.00
Echo	**1.00 1.00/0.00**	**1.00**	0.93 0.89/0.03	0.98
Pink	**1.00 1.00/0.00**	**1.00**	0.88 0.81/0.05	0.93
White	**0.91 0.86/0.04**	**0.95**	0.50 0.54/0.54	0.50
Fast (1.25x)	**0.99 0.99/0.00**	**1.00**	0.50 0.01/0.00	0.15
Smooth	**0.99 0.99/0.00**	1.00	0.94 0.93/0.04	0.98
Resample	1.00 1.00/0.00	1.00	1.00 1.00/0.00	1.00
AAC	1.00 1.00/0.00	1.00	1.00 1.00/0.00	1.00
MP3	**1.00 1.00/0.00**	**1.00**	1.00 0.99/0.00	0.99
EnCodec	**0.98 0.98/0.01**	**1.00**	0.51 0.52/0.50	0.50
Average	**0.96 0.98/0.04**	**0.97**	0.85 0.85/0.14	0.84

9.4.2 Comparison with watermarking

We evaluate the robustness of the detection on a wide range of audio editing operations: time modification (faster, resample), filtering (bandpass, highpass, lowpass), audio effects (echo, boost audio, duck audio), noise (pink noise, random noise), and compression (MP3, AAC, EnCodec). These attacks cover a wide range of transformations that are commonly used in audio editing software. For all edits except EnCodec compression, evaluation with parameters in the training range would be perfect. In order to show generalization, we chose stronger parameter to the attacks than those used during training (details in App. 9.9.2).

Detection is done on 10k ten-seconds audios from our VoxPopuli validation set. For each edit, we first build a balanced dataset made of the 10k watermarked/ 10k non-watermarked edited audio clips. We quantify the performance by adjusting the threshold of the detection score, selecting the value that maximizes accuracy (we provide corre-

sponding TPR and FPR at this threshold). The ROC AUC (Area Under the Curve of the Receiver Operating Characteristics) gives a global measure of performance over all threshold levels, and captures the TPR/FPR trade-off. To adapt data-hiding methods (*e.g.* WavMark) for proactive detection, we embed a binary message (chosen randomly beforehand) in the generated speech before release. The detection score is then computed as the Hamming distance between the original message and the one extracted from the scrutinized audio.

We observe in Tab. 9.3 that AudioSeal is overall more robust, with an average AUC of 0.97 vs. 0.84 for WavMark. The performance for lowpass and highpass filters indicates that AudioSeal embeds watermarks neither in the low nor in the high frequencies (WavMark focuses on high frequencies). We give results on more augmentations in App. 9.8.5.

Generalization. We evaluate how AudioSeal generalizes on various domains and languages. Specifically, we use the datasets ASVspoof (Liu et al., 2023d) and FakeAVCeleb (Khalid et al., 2021). Additionally, we translate speech samples from a subset of the Expresso dataset (Nguyen et al., 2023) (studio-quality recordings) using the SeamlessExpressive translation model (Seamless Communication et al., 2023). We select four target languages: Mandarin Chinese (CMN), French (FR), Italian (IT), and Spanish (SP). We also evaluate on non-speech AI-generated audios: music from MusicGen (Copet et al., 2024) and environmental sounds from AudioGen (Kreuk et al., 2023). Results are very similar to our in-domain test set and can be found in App. 9.8.4.

9.4.3 Localization

We evaluate localization with the sample-level detection accuracy, *i.e.* the proportion of correctly labeled samples, and the Intersection over Union (IoU). The latter is defined as the intersection between the predicted and the ground truth detection masks (1 when watermarked, 0 otherwise), divided by their union. IoU is a more relevant evaluation of the localization of short watermarks in a longer audio.

This evaluation is carried out on the same audio clips as for detection. For each one of them, we watermark a randomly placed segment of varying length. Localization with WavMark is a brute-force detection: a window of 1s slides over the 10s of speech with the default shift value of 0.05s. The Hammning distance between the 16 pattern bits is used

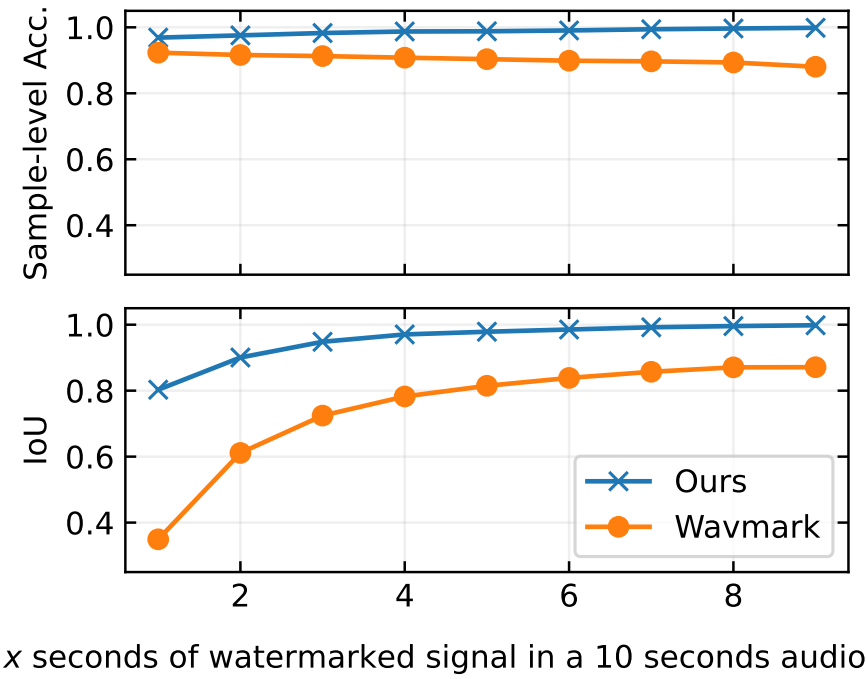

Figure 9.5 – Localization results across different durations of watermarked audio signals in terms of Sample-Level Accuracy and Intersection Over Union (IoU) metrics (↑ is better).

as the detection score. Whenever a window triggers a positive, we label its 16k samples as watermarked in the detection mask in $\{0,1\}^t$.

Figure 9.5 plots the sample-level accuracy and IoU for different proportions of watermarked speech in the audio clip. AudioSeal achieves an IoU of 0.99 when just one second of speech is AI-manipulated, compared to WavMark's 0.35. Moreover, AudioSeal allows for precise detection of minor audio alterations: it can pinpoint AI-generated segments in audio down to the sample level (usually 1/16k sec), while the concurrent WavMark only provides one-second resolution and therefore lags behind in terms of IoU. This is especially relevant for speech samples, where a simple word modification may greatly change meaning.

9.4.4 Attribution

Given an audio clip, the objective is now to find if any of N versions of our model generated it (detection), and if so, which one (identification). For evaluation, we create $N' = 100$ random 16-bits messages and use them to watermark 1k audio clips, each

Table 9.4 – Attribution results. We report the accuracy of the attribution (Acc.) and false attribution rate (FAR). Detection is done at FPR=10^{-3} and attribution matches the decoded message to one of N versions. We report averaged results over the edits of Tab. 9.3.

	N	1	10	10^2	10^3	10^4
FAR (%) ↓	WavMark	0.0	**0.20**	**0.98**	**1.87**	**4.02**
	AudioSeal	0.0	2.52	6.83	8.96	11.84
Acc. (%) ↑	WavMark	58.4	58.2	57.4	56.6	54.4
	AudioSeal	**68.2**	**65.4**	**61.4**	**59.3**	**56.4**

consisting of 5 seconds of speech (not 10s to reduce compute needs). This results in a total of 100k audios. For WavMark, the first 16 bits (/32) are fixed and the detection score is the number of well decoded pattern bits, while the second half of the payload hides the model version. An audio clip is flagged if the average output of the detector exceeds a threshold, corresponding to FPR=10^{-3}. Next, we calculate the Hamming distance between the decoded watermark and all N original messages. The message with the smallest Hamming distance is selected. It's worth noting that we can simulate $N > N'$ models by adding extra messages. This may represent versions that have not generated any sample.

False Attribution Rate (FAR) is the fraction of wrong attribution *among the detected audios* while the attribution accuracy is the proportion of detections followed by a correct attributions *over all audios*. AudioSeal has a higher FAR but overall gives a better accuracy, which is what ultimately matters. In summary, decoupling detection and attribution achieves better detection rate and makes the global accuracy better, at the cost of occasional false attributions.

9.4.5 Efficiency Analysis

To highlight the efficiency of AudioSeal, we conduct a performance analysis and compare it with WavMark. We apply the watermark generator and detector of both models on a dataset of 500 audio segments ranging in length from 1 to 10 seconds, using a single Nvidia Quadro GP100 GPU. The results are displayed in Fig. 9.6 and Tab. 9.5. In terms of generation, AudioSeal is 14x faster than WavMark. For detection, AudioSeal outperforms WavMark with two orders of magnitude faster performance on average, notably 485x faster in scenarios where there is no watermark (Tab. 9.5). This remarkable speed

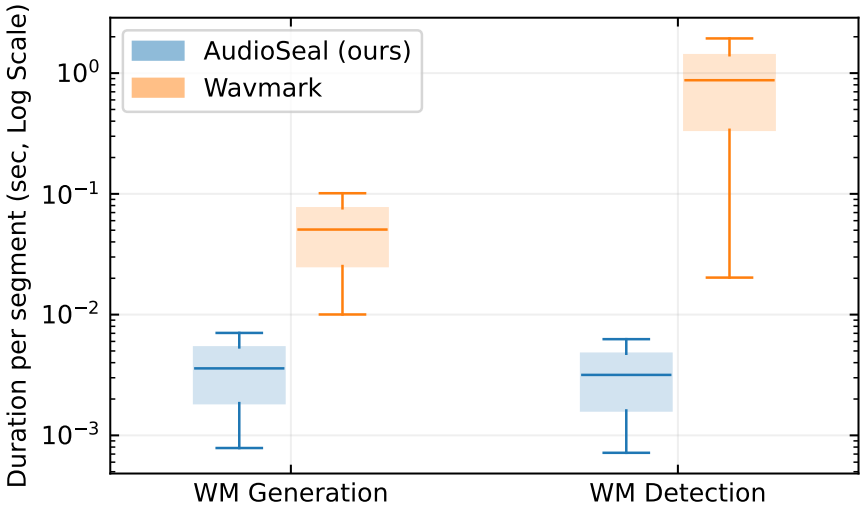

Figure 9.6 – Mean runtime (↓ is better). AudioSeal is one order of magnitude faster for watermark generation and two orders of magnitude faster for watermark detection for the same audio input. See Appendix 9.8.1 for full comparison.

increase is due to our model's unique localized watermark design, which bypasses the need for watermark synchronization (recall that WavMark relies on 20 pass forwards for a one-second snippet). AudioSeal's detector provides detection logits for each input sample directly with only one pass to the detector, significantly enhancing the detection's computational efficiency. This makes our system highly suitable for real-time and large-scale applications.

9.5 Adversarial Watermark Removal

We now examine more damaging deliberate attacks, where attackers might either "forge" the watermark by adding it to authentic samples (to overwhelm detection systems) or "remove" it to avoid detection. Our findings suggest that in order to maintain the effectiveness of watermarking against such adversaries, the code for training watermarking models and the awareness that published audios are watermarked can be made public. However, the detector's weights should be kept confidential.

We focus on watermark-removal attacks and consider three types of attacks depending on the adversary's knowledge:

— *White-box*: the adversary has access to the detector (*e.g.* because of a leak), and

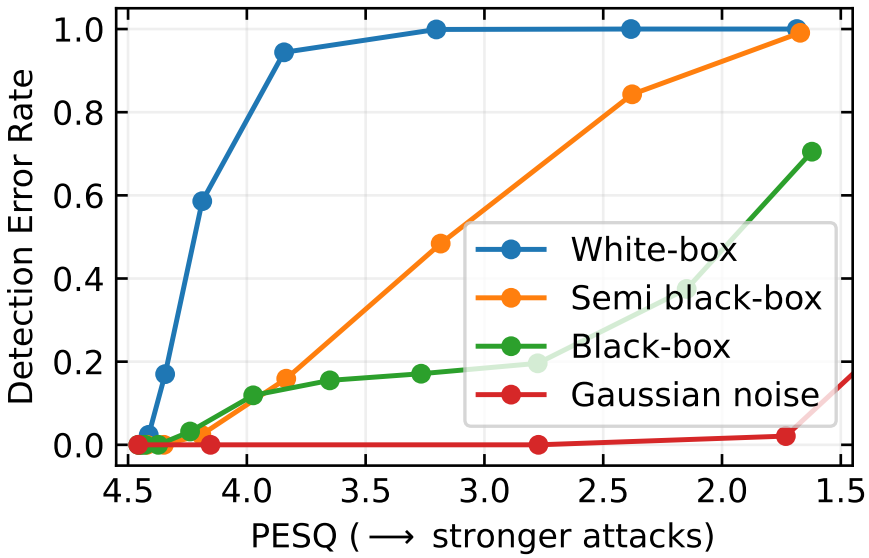

Figure 9.7 – Watermark-removal attacks. PESQ is measured between attacked audios and genuine ones (PESQ < 4 strongly degrades the audio quality). The more knowledge the attacker has over the watermarking algorithm, the better the attack is.

performs a gradient-based adversarial attack against it. The optimization objective is to minimize the detector's output.

— *Semi black-box*: the adversary does not have access to any weights, but is able to re-train generator/detector pairs with the same architectures on the same dataset. They perform the same gradient-based attack as before, but using the new detector as proxy for the original one.

— *Black-box*: the adversary does not have any knowledge on the watermarking algorithm being used, but has access to an API that produces watermarked samples, and to negative speech samples from any public dataset. They first collect samples and train a classifier to discriminate between watermarked and not-watermarked. They attack this classifier as if it were the true detector.

For every scenario, we watermark 1k samples of 5 seconds, then attack them. The gradient-based attack optimizes an adversarial noise added to the audio, with 100 steps of Adam. During the optimization, we control the norm of the noise to trade off attack strength and audio quality. When training the classifier for the black-box attack, we use 80k/80k watermarked/genuine samples of 8 seconds and make sure the classifier has 100% detection accuracy on the validation set. More details in App. 9.9.5.

Figure 9.7 contrasts various attacks at different intensities, using Gaussian noise as a

reference. The white-box attack is by far the most effective one, increasing the detection error by around 80%, while maintaining high audio quality (PESQ > 4). Other attacks are less effective, requiring significant audio quality degradation to achieve 50% increase the detection error, though they are still more effective than random noise addition. In summary, the more is disclosed about the watermarking algorithm, the more vulnerable it is. The effectiveness of these attacks is limited as long as the detector remains confidential.

9.6 Conclusion

In this chapter, we introduced AudioSeal, a proactive method for the detection, localization, and attribution of AI-generated speech. AudioSeal revamps the design of audio watermarking to be specific to localized detection rather than data hiding. It is based on a generator/detector architecture that can generate and extract watermarks at the audio sample level. This removes the dependency on slow brute force algorithms, traditionally used to encode and decode audio watermarks. The networks are jointly trained through a novel loudness loss, differentiable augmentations and masked sample level detection losses. As a result, AudioSeal achieves state-of-the-art robustness to various audio editing techniques, very high precision in localization, and orders of magnitude faster runtime than methods relying on synchronization. Through an empirical analysis of possible adversarial attacks, we conclude that for watermarking to still be an effective mitigation, the detector's weights have to be kept private – otherwise adversarial attacks might be easily forged. A key advantage of AudioSeal is its practical applicability. It stands as a ready-to-deploy solution for watermarking in voice synthesis APIs. This is pivotal for large-scale content provenance on social media and for detecting and eliminating incidents, enabling swift action on instances like the US voters' deepfake case (Murphy et al., 2024) long before they spread.

Impact Statement

This research aims to improve transparency and traceability in AI-generated content, but watermarking in general can have a set of potential misuses such as government surveillance of dissidents or corporate identification of whistle blowers. Additionally, the watermarking technology might be misused to enforce copyright on user-generated content, and its ability to detect AI-generated audio could increase skepticism about digital

communication authenticity, potentially undermining trust in digital media and AI. However, despite these risks, ensuring the detectability of AI-generated content is important, along with advocating for robust security measures and legal frameworks to govern the technology's use.

Table 9.5 – The average runtime (ms) per sample of our proposed AudioSeal model against the state-of-the-art Wavmark(Chen et al., 2024a) method. Our experiments were conducted on a dataset of audio segments spanning 1 sec to 10 secs, using a single Nvidia Quadro GP100 GPU. The results, displayed in the table, demonstrate substantial speed enhancements for both Watermark Generation and Detection with and without the presence of a watermark. Notably, for watermark detection, AudioSeal is **485× faster** than Wavmark during the absence of a watermark, more details in section 9.4.5.

Model	Watermarked	**Detection ms (speedup)**	**Generation ms (speedup)**
Wavmark	No	1710.70 ± 1314.02	–
AudioSeal (ours)	No	**3.25 ± 1.99** (**485×**)	–
Wavmark	Yes	106.21 ± 66.95	104.58 ± 65.66
AudioSeal (ours)	Yes	**3.30 ± 2.03** (**35×**)	**7.41 ± 4.52** (**14 ×**)

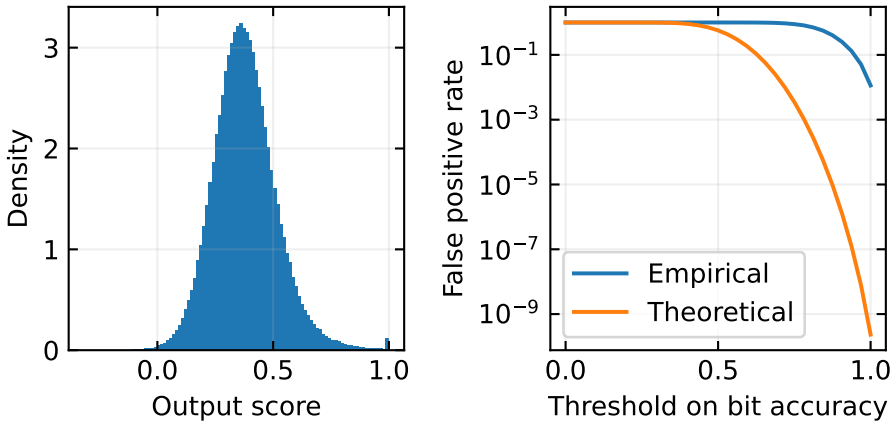

Figure 9.8 – (Left) Histogram of scores output by WavMark's extractor on 10k genuine samples. (Right) Empirical and theoretical FPR when the chosen hidden message is all 0.

AudioSeal Appendix

9.7 False Positive Rates - Theory and Practice

Theoretical FPR. When doing multi-bit watermarking, previous works (Chen et al., 2024a; Fernandez et al., 2023; Kim et al., 2024; Yu et al., 2021) usually extract the message m' from the content x and compare it to the original binary signature $m \in \{0,1\}^k$ embedded in the speech sample. The detection test relies on the number of matching bits $M(m, m')$:

$$\text{if } M\left(m, m'\right) \geq \tau \quad \text{where} \quad \tau \in \{0, \ldots, k\}, \tag{9.4}$$

then the audio is flagged. This provides theoretical guarantees over the false positive rates.

Formally, the statistical hypotheses are H_1: "The audio signal x is watermarked", and the null hypothesis H_0: "The audio signal x is genuine". Under H_0 (*i.e.*, for unmarked audio), if the bits $m'_1, \ldots, m'_k$ are independent and identically distributed (i.i.d.) Bernoulli random variables with parameter 0.5, then $M(m, m')$ follows a binomial distribution with parameters (k, 0.5). The False Positive Rate (FPR) is defined as the probability that $M(m, m')$ exceeds a given threshold τ. A closed-form expression can be given using the regularized incomplete beta function $I_x(a; b)$ (linked to the CDF of the binomial distribution):

$$\mathrm{FPR}(\tau) = \mathbb{P}\left(M \geq \tau | H_0\right) = I_{1/2}(\tau, k - \tau + 1). \tag{9.5}$$

Empirical study. We empirically study the FPR of WavMark-based detection on our validation dataset. We use the same parameters as in the original paper, *i.e.* $k = 32$-bits are extracted from 1s speech samples. We first extract the soft bits (before thresholding) from 10k genuine samples and plot the histogram of the scores in Fig. 9.8 (left). We should observe a Gaussian distribution with mean 0.5, while empirically the scores are centered around 0.38. This makes the decision heavily biased towards bit 0 on genuine samples. It is therefore impossible to theoretically set the FPR since this would largely underestimate the actual one. For instance, Figure 9.8 (right) shows the theoretical and empirical FPR for different values of τ when the chosen hidden message is full 0. Put differently, the argument that says that hiding bits allows for theoretical guarantees over the detection rates is not valid in practice.

9.8 Additional Experimental Results

9.8.1 Computational efficiency

We show in Figure 9.9 the mean runtime of the detection and generation depending on the audio duration. Corresponding numbers are given in Table 9.5.

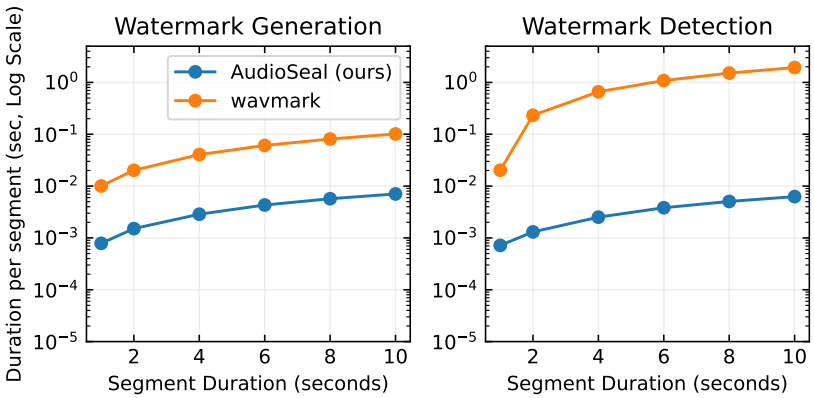

Figure 9.9 – **Mean runtime** (↓ is better) of AudioSeal versus WavMark. AudioSeal is one order of magnitude faster for watermark generation andtwo orders of magnitude faster for watermark detection for the same audio input, signifying a considerable enhancement in real-time audio watermarking efficiency.

9.8.2 Another architecture

Our architecture relies on the SOTA compression method EnCodec. However, to further validate our approach, we conduct an ablation study using a different architecture DPRNN (Luo et al., 2020). The results are presented in Tab. 9.6. They show that the performance of AudioSeal is consistent across different architectures, with similar performances using the much slower and heavier architecture from Luo et al. (2020). This indicates that model capacity is not a limiting factor for AudioSeal.

Table 9.6 – Results of AudioSeal with different architectures for the generator and detector. The IoU is computed for 1s of watermark in 10s audios (corresponding to the leftmost point in Fig. 9.5).

Method	SISNR	STOI	PESQ	Acc.	IoU
EnCodec	26.00	0.997	4.470	1.00	0.802
DPRNN	26.7	0.996	4.421	1.00	0.796

9.8.3 Audio mixing

We hereby evaluate the scenario where two watermarked signals (e.g., vocal and instrumental) are mixed together. To explore this, we conducted experiments using a non-vocal music dataset. In these experiments, we normalized and summed the loudness of watermarked speech and music segments. The results are detailed Tab. 9.7.

Table 9.7 – Detection results for watermarked speech and music mixed signals. ✓ and ✗ indicate the presence of the watermark.

Speech	BG Music	Acc. FPR / TPR	AUC
✓	✓	0.9996 0.0003 / 0.9996	0.9999
✓	✗	0.9787 0.0310 / 0.9883	0.9961

9.8.4 Out of domain (OOD) evaluations

Table 9.8 – Evaluation of AudioSeal Generalization across domains and languages. Namely, translations of speech samples from the Expresso dataset (Nguyen et al., 2023) to four target languages: Mandarin Chinese (CMN), French (FR), Italian (IT), and Spanish (SP), using the SeamlessExpressive model (Seamless Communication et al., 2023). Music from MusicGen (Copet et al., 2024) and environmental sounds from AudioGen (Kreuk et al., 2023).

Aug	Seam. (Cmn)	Seam. (Spa)	Seam. (Fra)	Seam.(Ita)	Seam. (Deu)	Voicebox (Eng)	AudioGen	MusicGen
None	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Bandpass	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Highpass	0.71	0.68	0.70	0.70	0.70	0.64	0.52	0.52
Lowpass	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00
Boost	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Duck	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Echo	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Pink	0.99	1.00	0.99	1.00	0.99	1.00	1.00	1.00
White	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Fast	0.97	0.98	0.99	0.98	0.99	0.98	0.87	0.87
Smooth	0.96	0.99	0.99	0.99	0.99	0.99	0.98	0.98
Resample	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
AAC	0.99	0.99	0.99	0.99	0.99	0.97	0.99	0.98
MP3	0.99	0.99	0.99	0.99	0.99	0.97	0.99	1.00
Encodec	0.97	0.98	0.99	0.99	0.98	0.96	0.95	0.95
Average	0.97	0.97	0.98	0.98	0.98	0.97	0.95	0.95

As previously outlined in Sec. 9.4.2, we tested AudioSeal on the outputs of various voice cloning models and other audio modalities. We employed the same set of augmentations and observed very similar results, as demonstrated in Tab. 9.8. Interestingly, even though we did not train our model on AI-generated speech, we noticed an improvement in performance compared to our test data. No sample was misclassified among the 10k samples that comprised each of our out-of-distribution (OOD) datasets. We also provide the other perceptual metrics results on OOD data in Tab. 9.9.

We also evaluated AudioSeal on three additional datasets containing real human speech: AudioSet (Gemmeke et al., 2017b), ASVspoof (Liu et al., 2023d), and FakeAVCeleb (Khalid et al., 2021). Again, we observed similar performance, as shown in Tab. 9.10.

Table 9.9 – Audio quality and intelligibility evaluations on AI generated speech data from various models and languages.

Model	Dataset	SISNR	PESQ	STOI	VISQOL
AudioSeal	Seam. (Deu)	23.35	4.244	0.999	4.688
	Seam. (Fr)	24.02	4.199	0.998	4.669
	Voicebox	25.23	4.449	0.998	4.800
WavMark	Seam. (Deu)	38.93	3.982	0.999	4.515
	Seam. (Fr)	39.06	3.959	0.999	4.506
	Voicebox	39.63	4.211	0.998	4.695

Table 9.10 – Evaluation of the detection performances on different datasets. AudioSet is an environmental sounds dataset while ASVspoof (Liu et al., 2023d) and FakeAVCeleb (Khalid et al., 2021) are deep-fake detection datasets.

Dataset	Acc. TPR/FPR	AUC
Audioset	0.9992 0.9996/0.0011	1.0
ASVspoof	1.0 1.0/0.0	1.0
FakeAVCeleb	1.0 1.0/0.0	1.0

9.8.5 Robustness results

We plot the detection accuracy against the strength of multiple augmentations in Fig. 9.10. AudioSeal outperforms WavMark for most augmentations at the same strength. However, for highpass filters above our training range (500Hz) WavMark has a much better detection accuracy. Our system's TF-loudness loss embeds the watermark where human speech carries the most energy, typically lower frequencies, due to auditory masking. This contrasts with WavMark, which places the watermark in higher frequency bands. Embedding the watermark in lower frequencies is advantageous. For example, speech remains audible with a lowpass filter at 1500 Hz, but not with a highpass filter at the same frequency. This difference is measurable with PESQ in relation to the original audio, making it more beneficial to be robust against a lowpass filter at a 1500 Hz cut-off than a highpass filter at the same cut-off:

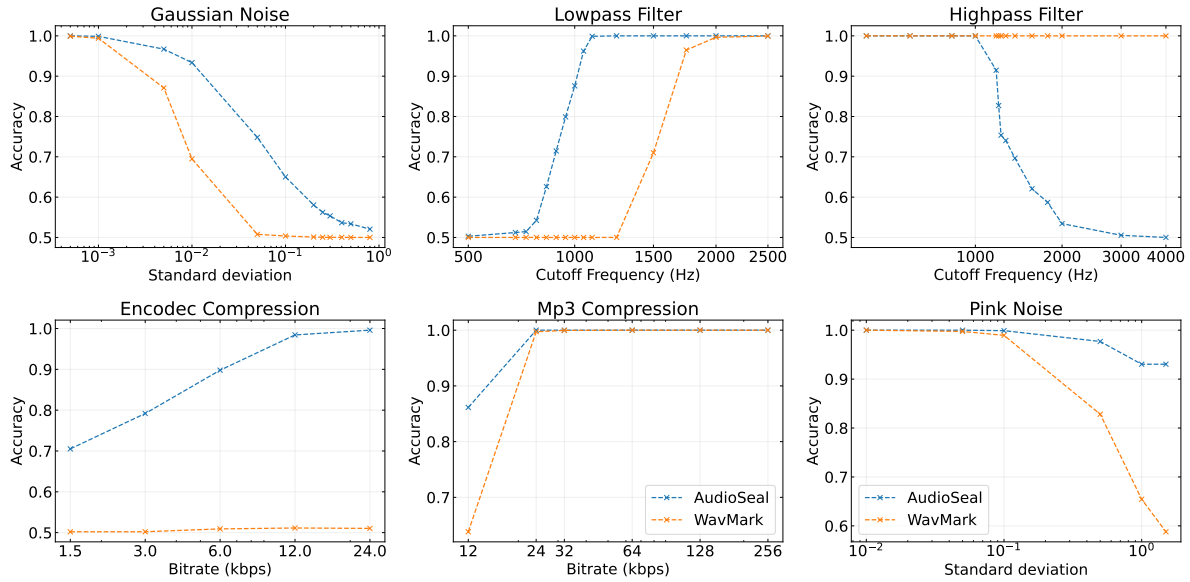

Figure 9.10 – Accuracy of the detector on augmented samples with respect to the strength of the augmentation.

Filter Type	PESQ	AudioSeal	WavMark
Highpass 1500Hz	1.85 ✗	0.7	1.0
Lowpass 1500Hz	2.93 ✓	1.0	0.7

9.9 Experimental details

9.9.1 Loudness

Our loudness function is based on a simplification of the implementation in the torchaudio (Yang et al., 2022) library. It is computed through a multi-step process. Initially, the audio signal undergoes K-weighting, which is a filtering process that emphasizes certain frequencies to mimic the human ear's response. This is achieved by applying a treble filter and a highpass filter. Following this, the energy of the audio signal is calculated for each block of the signal. This is done by squaring the signal and averaging over each block. The energy is then weighted according to the number of channels in the audio signal, with different weights applied to different channels to account for their varying contributions to perceived loudness. Finally, the loudness is computed by taking the logarithm of the weighted sum of energies and adding a constant offset.

9.9.2 Robustness Augmentations

Here are the details of the audio editing augmentations used at train time (T), and evaluation time (E):

— **Bandpass Filter:** Combines highpass and lowpass filtering to allow a specific frequency band to pass through. (T) fixed between 300Hz and 8000Hz; (E) fixed between 500Hz and 5000Hz.

— **Highpass Filter:** Uses a highpass filter on the input audio to cut frequencies below a certain threshold. (T) fixed at 500Hz; (E) fixed at 1500Hz.

— **Lowpass Filter:** Applies a lowpass filter to the input audio, cutting frequencies above a cutoff frequency. (T) fixed at 5000Hz; (E) fixed at 500Hz.

— **Speed:** Changes the speed of the audio by a factor close to 1. (T) random between 0.9 and 1.1; (E) fixed at 1.25.

— **Resample:** Upsamples to intermediate sample rate and then downsamples the audio back to its original rate without changing its shape. (T) and (E) 32kHz.

— **Boost Audio:** Amplifies the audio by multiplying by a factor. (T) factor fixed at 1.2; (E) fixed at 10.

— **Duck Audio:** Reduces the volume of the audio by a multiplying factor. (T) factor fixed at 0.8; (E) fixed at 0.1.

— **Echo:** Applies an echo effect to the audio, adding a delay and less loud copy of the original. (T) random delay between 0.1 and 0.5 seconds, random volume between 0.1 and 0.5; (E) fixed delay of 0.5 seconds, fixed volume of 0.5.

— **Pink Noise:** Adds pink noise for a background noise effect. (T) standard deviation fixed at 0.01; (E) fixed at 0.1.

— **White Noise:** Adds gaussian noise to the waveform. (T) standard deviation fixed at 0.001; (E) fixed at 0.05.

— **Smooth:** Smooths the audio signal using a moving average filter with a variable window size. (T) window size random between 2 and 10; (E) fixed at 40.

— **AAC:** Encodes the audio in AAC format. (T) bitrate of 128kbps; (E) bitrate of 64kbps.

— **MP3:** Encodes the audio in MP3 format. (T) bitrate of 128kbps; (E) bitrate of 32kbps.

— **EnCodec:** Resamples at 24kHz, encodes the audio with EnCodec with $nq = 16$ (16 streams of tokens), and resamples it back to 16kHz.

Implementation is done with the `julius` python library.

9.9.3 Networks architectures (Fig. 9.3)

The watermark generator is composed of an encoder and a decoder, both incorporating elements from EnCodec (Défossez et al., 2023). The encoder applies a 1D convolution with 32 channels and a kernel size of 7, followed by four convolutional blocks. Each of these blocks includes a residual unit and down-sampling layer, which uses convolution with stride S and kernel size $K = 2S$. The residual unit has two kernel-3 convolutions with a skip-connection, doubling channels during down-sampling. The encoder concludes with a two-layer LSTM and a final 1D convolution with a kernel size of 7 and 128 channels. Strides S values are (2, 4, 5, 8) and the nonlinear activation in residual units is the Exponential Linear Unit (ELU). The decoder mirrors the encoder but uses transposed convolutions instead, with strides in reverse order.

The detector comprises an encoder, a transposed convolution and a linear layer. The encoder shares the generator's architecture (but with different weights). The transposed convolution has h output channels and upsamples the activation map to the original audio resolution (resulting in an activation map of shape (t, h)). The linear layer reduces the h dimensions to two, followed by a softmax function that gives sample-wise probability scores.

9.9.4 MUSHRA protocole detail

The MUSHRA protocol is a crowdsourced test in which participants rate the quality of various samples on a scale of 0 to 100. The ground truth is provided for reference. We utilized 100 speech samples, each lasting 10 seconds. Each sample was evaluated by at least 20 participants. As part of the study, we included a low anchor, which is a very lossy compression at 1.5kbps, encoded using EnCodec. Participants who failed to assign the lowest score to the low anchor for at least 80% of their assignments were excluded from the study. For comparison, the ground truth samples received an average score of 80.49, while the low anchor's average score was 53.21.

9.9.5 Attacks on the watermark

Adversarial attack against the detector. Given a sample x and a detector D, we want to find $x' \sim x$ such that $D(x') = 1 - D(x)$. To that end, we use a gradient-based attack. It starts by initializing a distortion δ_{adv} with random gaussian noise. The algorithm iteratively updates the distortion for a number of steps n. For each step, the distortion is added to the original audio via $x = x + \alpha.\tanh(\delta_{adv})$, passed through the model to get predictions. A cross-entropy loss is computed with label either 0 (for removal) or 1 (for forging), and back-propagated through the detector to update the distortion, using the Adam optimizer. At the end of the process, the adversarial audio is $x + + \alpha.\tanh(\delta_{adv})$. In our attack, we use a scaling factor $\alpha = 10^{-3}$, a number of steps $n = 100$, and a learning rate of 10^{-1}. The tanh function is used to ensure that the distortion remains small, and gives an upper bound on the SNR of the adversarial audio.

Training of the malicious detector. Here, we are interested in training a classifier that can distinguish between watermarked and non-watermarked samples, when access to many samples of both types is available. To train the classifier, we use a dataset made of more than 80k samples of 8 seconds speech from Voicebox (Le et al., 2024) watermarked using our proposed method and a similar amount of genuine (un-watermarked) speech samples. The classifier shares the same architecture as AudioSeal's detector. The classifier is trained for 200k updates with batches of 64 one-second samples. It achieves perfect classification of the samples. This is coherent with the findings of Voicebox (Le et al., 2024).

Chapter 10

Safe and Open-Sourceable Audio LMs with Latent Watermarking

This chapter describes a method that builds on AudioSeal (San Roman et al., 2024) to achieve in-model watermarking with Audio Language Models. It is based on the paper "Latent Watermarking of Audio Generative Models" (San Roman et al., 2025) published at ICASSP 2025.

10.1 Introduction

Sophisticated generative models are impacting various audio modalities: environmental sounds (Kreuk et al., 2023; Liu et al., 2023c), music (Agostinelli et al., 2023; Copet et al., 2024), and speech (Borsos et al., 2023; Vyas et al., 2023; Wang et al., 2023a). These models produce outputs increasingly indistinguishable from real data (OpenAI et al., 2024; Rombach et al., 2021). Their rapid proliferation and quality raise concerns about misuse (e.g. creation of deepfakes) and respect for intellectual property. These concerns are heightened when models are open-sourced, since they can be easily accessed and used by anyone, including malicious actors. Consequently, regulators suggest watermarking to label and detect generative model outputs (refer to the EU AI Act, White House executive order, and CAC measures).

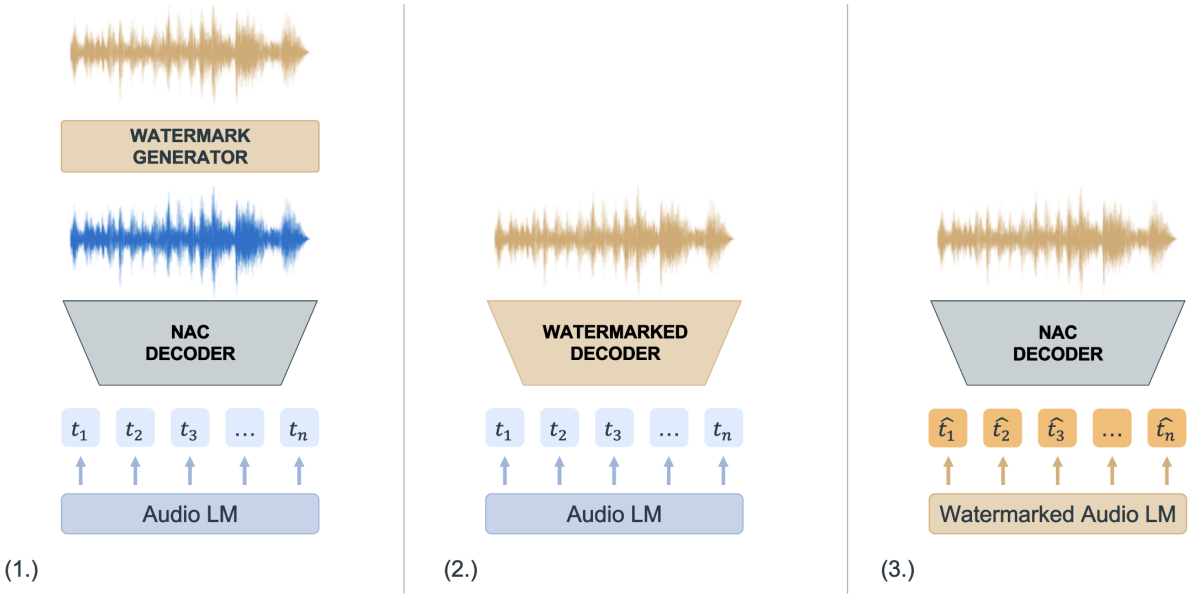

Figure 10.1 – Watermarking for Audio Language Models at different stages. (1.) Post-hoc watermarking: the audio is watermarked at the end of the generation process, after the audio is generated. (2.) In-decoder watermarking: the watermark is embedded when decoding audio tokens, similar to stable signature (Fernandez et al., 2023). (3.) (ours) In-LM watermarking: the latent audio tokens are already watermarked when they are generated by the audio language model.

Watermarking is a technique that slightly alters the audio after its generation, in a way that is inaudible for humans but identifiable by specific detection algorithms. The state-of-the-art methods are based on deep neural networks (Chen et al., 2024a; San Roman et al., 2024) that are trained end-to-end to embed and detect watermarks in audio signals, even after audio compression or editing. Such methods are for instance employed to safeguard APIs for public model demonstrations (Seamless Communication et al., 2023; Vyas et al., 2023). However, while post-hoc watermarking has proven effective in certain scenarios, it is not as effective for protecting open-sourced models, as malicious users could potentially extract the output before the watermarking stage (for example by commenting out the code responsible for watermark embedding).

In the image domain, some methods (Fernandez et al., 2023; Kim et al., 2024) fine-tune decoders to output watermarked images directly, to make it compliant with open-sourcing. However, in the audio domain, it is common and cost-effective to train decoders (also called vocoders) that convert latent representations to waveforms (Kong et al., 2020; Kumar et al., 2019; San Roman et al., 2023). Watermarking can thus be easily bypassed

by using non-watermarked vocoders. Therefore, in this chapter, we propose to watermark the latent generative model that creates the latent representations. Figure 10.1 illustrates how watermarking can be applied at different stages of the generation process.

We focus on MusicGen (Copet et al., 2024) due to its performance and adoption. It consists of an auto-encoder EnCodec (Défossez et al., 2023) that compresses audio into discrete representations (tokens) and a single-stage transformer (audio Language Model, LM) that predicts the next tokens and decodes them into a music stream. We train watermark generator/detector models to be robust to EnCodec. Intuitively, this makes both the audio and encoded tokens watermarked. We then train the LM on tokens derived from audios that were preemptively marked. The resulting LM produces tokens whose decoded outputs are watermarked, irrespective of the LM conditioning or decoding algorithm. In other terms, as long as the watermarking algorithm withstands the audio tokenization, the watermark transfers from the training data to the generative model outputs.

In short, (1) we introduce a way to watermark audio generative models at the latent representations level, (2) we demonstrate that it makes generations detectable with high confidence while having almost no influence on the model performance, (3) we demonstrate the robustness of the watermark to model-level changes, namely, switching the decoding algorithm and fine-tuning the audio LM.

10.2 Audio Model Watermarking

10.2.1 Problem statement

We consider providers training an audio-generative model on a large proprietary dataset. They aim to release the model publicly but worry about misuse and unauthorized redistribution. To mitigate these concerns, they watermark the model during training to enhance the detection of generated content or unauthorized API usage. We describe this watermarking process below and provide a step by step overview in Figure 9.1.

10.2.2 Audio watermarking

We first build an audio watermarking model based on AudioSeal (San Roman et al., 2024). It jointly trains a watermark generator G and a watermark detector D. G takes a

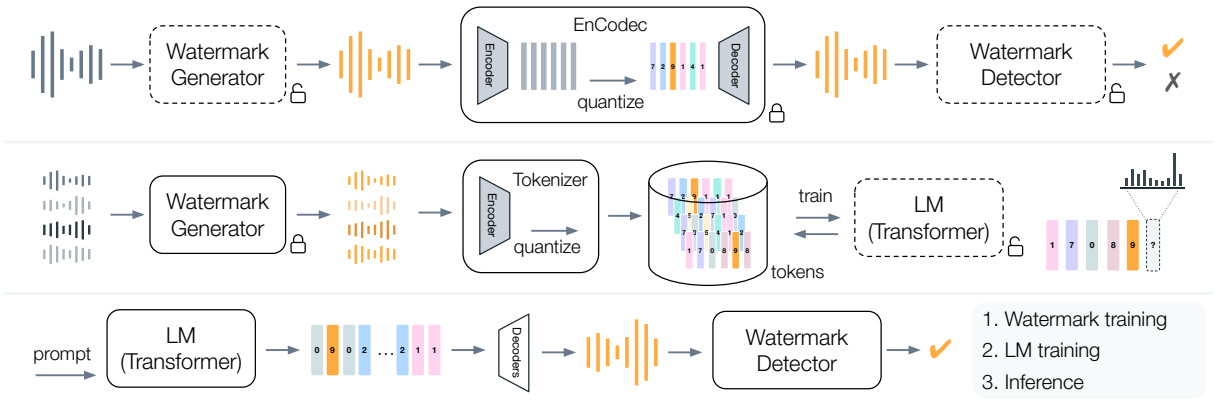

Figure 10.2 – Overview of our method. (1.) We train a watermark generator and detector based on AudioSeal (San Roman et al., 2024), enhancing robustness against EnCodec (Défossez et al., 2023) by processing the watermarked audio through EnCodec before detection. (2.) We watermark the audios from our database and train a MusicGen (Copet et al., 2024) model for next token prediction on this watermarked data. (3.) During inference, we prompt (using text or audio) the language model and decode audios that are detectable with watermark detector.

signal $s \in \mathbb{R}^T$ and generates an additive watermark δ_w, that is made imperceptible through perceptual losses $\mathcal{L}_{\text{percep}}(s, s + \delta_w)$. The watermarked audio $s + \delta_w$ is augmented into s'. Augmentations include padding the audio with 0, replacing intervals of watermarked audio with non-watermarked audio from the same batch, or dropping the δ_w. s' is fed to the detector, which is trained to output which part of a waveform is watermarked via a localization loss $\mathcal{L}_{\text{loc}}(D(s'), y')$, where $y' \in \{0, 1\}^T$ indicates the watermarked intervals in s' (1 for watermarked, 0 otherwise).

We make the following changes with regard to AudioSeal's recipe. We remove the message encoder to only focus on watermark detection. Furthermore, it is important to remember that the audio LM will be trained on tokens, not directly on audio. Hence, the LM will not retain watermark information if it is absent at the discrete representation level. Therefore, we train the watermark generation/detection to be very robust to the specific EnCodec compression model later used for audio tokenization (see Sec. 10.2.3). This is done by oversampling this EnCodec augmentation so that 50% of batches go through EnCodec before the detection phase.

10.2.3 Audio language model

We select MusicGen (Copet et al., 2024) as the audio LM to watermark.

Watermarking. The first step is to watermark the audios with the model of Sec. 10.2.2. This is done on the fly at loading time (this takes around 10 ms for a 10-second audio).

Tokenization. We use the EnCodec compression model to transform the audio signal into a discrete sequence of tokens suitable for language modeling. It uses residual vector quantization (RVQ) (Zeghidour et al., 2021) which compresses an audio signal $s \in \mathbb{R}^T$ into K streams of tokens $(u_i^{(j)})_{j\in[1,K]\,;\,i\in[0,T/f_r]}$ (f_r being the frame rate). This model is trained on audio segments sampled at a rate of 32 kHz and $f_r = 50$ Hz, the number of codebooks is $K = 4$ and the codebook size is 2048 ($u_i^{(j)} \in [1, 2048]$). Overall, this results in a bit rate of 2.2 kbit per second.

Language modeling aims to build a probabilistic model of sequences of discrete tokens. MusicGen implements a delay pattern (Kharitonov et al., 2022b) that adds a delay k to the k-th residual. It allows the model to generate the tokens in a coarse to fine order. This way all the streams can be sampled in parallel while assuring that all the previous residuals are fixed when sampling a given token. Put differently, the transformer is fed with a sequence of embeddings created from K tokens: $s_i = \{u_{i-4}^{(4)}, u_{i-2}^{(3)}, u_{i-1}^{(2)}, u_i^{(1)}\}$. The embedding of s_i is then the sum of the embedding of each of its constitutive tokens, with additional sinusoidal embeddings. As most current language models (OpenAI et al., 2024; Touvron et al., 2023), training is done with next token prediction and the cross-entropy is computed per codebook.

10.3 Experiments

10.3.1 Experimental details

Watermarking models. We train on an internal music dataset containing 1.5k songs at 32 kHz sample rate, with 1-second audio excerpts. The model is trained for 400k steps with batch size 64. Hyper-parameters (network architectures, optimizer, etc.) are kept the same as in the original work (San Roman et al., 2024).

MusicGen models. We use 20k hours of licensed music to train the models with two different model sizes: small (300M parameters) and medium (1.5B parameters). They are similar in quality and diversity to the ones used by Copet et al. (2024). We use public implementation and default parameters available on the AudioCraft GitHub page. Each

Table 10.1 – Subjective evaluation. We compare audio quality and text relevance of the original MusicGen models (ori.) and our models that natively outputs watermarked audios (ours).

Size	OVL (↑)	REL (↑)
Ground Truth	93.08 ± 0.53	93.01 ± 0.68
Small (ori.)	83.67 ± 1.85	82.42 ± 1.37
Small (ours)	$\mathbf{84.13} \pm 2.21$	$\mathbf{82.46} \pm 1.44$
Medium (ori.)	$\mathbf{85.92} \pm 1.46$	$\mathbf{83.71} \pm 1.79$
Medium (ours)	84.91 ± 1.53	82.64 ± 1.41

model is trained for 200 epochs with a batch size of 128, with the default optimizer. We use 64 GPUs for the medium model and 32 for the small.

Inference. For music generation sampling, we use top-k sampling with $k = 250$ tokens and a temperature of 1.0.

10.3.2 Quality of the audio generative model

We first subjectively evaluate how watermarking influences the quality of the generative models.

We adhere to the original paper's protocol (Copet et al., 2024), using (OVL) to assess sample quality and (REL) to evaluate relevance to text prompts. Models are tested on 15-second generation using 40 text prompts from the test set of MusicCaps (Agostinelli et al., 2023). Every sample is rated by 20 listeners that rate them on a scale from 1 to 100. For every study, we report both mean score and CI95. Table 10.1 shows that the performance difference between a model trained on watermarked data and one trained on normal data is negligible. This holds true for both sizes, with the rating difference falling within the confidence interval.

10.3.3 Detection and localization results

Detection. To evaluate the detection performance, we generate 10k positive 15-second samples with the watermarked model and use 10k negative samples from our test set that we compress using the codec model. The watermark detector gives a score per

Table 10.2 – Detection and localization results on 10k positive/negative samples, for the MusicGen models watermarked post-hoc using AudioSeal (+AS), or in-model with our method (ours). We report the area under the ROC curve (AUC), the accuracy for the best threshold, as well as the intersection over union (IoU) and sample level accuracy (SL-Acc.).

	Detection			Localization	
Model	AUC	Acc.	TPR / FPR	IoU	SL-Acc.
Small + AS	1.0	1.0	1.0 / 0.0	1.0	1.0
Medium + AS	1.0	1.0	1.0 / 0.0	1.0	1.0
Small (ours)	0.999	0.993	0.986 / 2.10^{-4}	0.81	0.91
Medium (ours)	0.999	0.994	0.988 / 3.10^{-4}	0.91	0.96

time-step of the audio, which we average to get a score for the whole audio. Audio is flagged as watermarked if this score is higher than a threshold τ. We report in Tab. 10.2 the area under the ROC curve, as well as the accuracy for the best τ (and true positive and false positive rates at this τ, TPR, and FPR). The generated output is indeed watermarked as indicated by the detection metrics: the AUC is close to 1 and TPR is higher than 0.95 at FPR around 10^{-4}.

Localization. We then evaluate if the detector still has the property to locally detect watermarked segments. To do so, we generate 15-second samples and replace parts with other non-watermarked audio from our test set. The proportion of signal that is watermarked is 50% on average. We then measure the precision of the detection using the detection accuracy at the sample level together with the intersection over union (IoU) metric. For localization, we use a fixed detection threshold set at 0.5. As shown in Tab. 10.2, results are on-par (although a bit lower) to AudioSeal, showing that the detector keeps a good-enough performance on the localization of generated outputs.

Robustness. We evaluate the robustness to different audio edits, and compare the performance of our in-model watermarking and of post-hoc watermarking that directly applies the watermark to generated outputs with AudioSeal. The evaluation is made on 10k samples.

Table 10.3 shows that while in-model watermarking keeps a decent robustness to common audio edits there is a slight performance decrease compared to the original watermarking model. Therefore, when post-hoc watermarking is feasible, it might be prefer-

Table 10.3 – Robustness to audio edits for post-hoc watermarking with AudioSeal or using our in-model method. The results are computed on audios generated with the Small model.

	Post-hoc		In-model	
Edit	AUC	Acc TPR / FPR	AUC	Acc TPR / FPR
White Noise	0.995	0.99 0.99/0.01	0.941	0.93 0.91/0.04
Lowpass	0.99	0.99 0.99/0.01	0.942	0.94 0.91/0.03
Highpass	0.931	0.98 0.98/0.2	0.941	0.93 0.91/0.04
Resample	0.940	0.94 0.91/0.03	0.999	0.99 0.99/0.01
AAC	0.999	0.98 0.97/0.01	0.88	0.81 0.81/0.04
Pink Noise	0.996	0.97 0.98/0.03	0.956	0.92 0.91/0.06
Echo	0.950	0.98 0.99/0.02	0.906	0.92 0.89/0.04

able to in-model watermarking. The latter is better suited for scenarios where post-hoc watermarking is not possible, such as when open-sourcing a model.

10.4 Attacks on the Model's Watermark

We now focus on model-level attacks, *i.e.*, modifications of the model attempting to make its outputs undetectable.

10.4.1 Switching decoder

Previous works alter the latent decoder to embed the watermark (Fernandez et al., 2023; Juvela and Wang, 2024a). However, audio vocoders are relatively easy to train and interchange (Kong et al., 2020; 2021; Kumar et al., 2019). They do not necessitate extensive data or computational power compared to those required for training an audio LM. Therefore, replacing the decoder to use the watermark-free generative model is rather straightforward. In contrast, our work embeds the watermark at the latent stage for robustness against decoder changes.

We now evaluate how the change of the decoder influences detection performance. In previous experiments, the default decoder was the codec model from MusicGen. We replace it with Multi-Band Diffusion (San Roman et al., 2023) which uses diffusion to map discrete EnCodec tokens into the waveform domain, and a discrete version HiFi-GAN (Kong et al., 2020), which we trained on tokens-waveform pairs. We use the same

Table 10.4 – Detection results with other decoding algorithm.

Decoder	AUC	Acc.	TPR / FPR
HiFi-GAN	0.999	0.990	0.980 / 4.10^{-4}
Multi-band diffusion	0.991	0.954	0.951 / 0.043
Default	0.999	0.993	0.986 / 0.000

experimental setup as in Sec. 10.3.3, but with different algorithms to decode the tokens. Table 10.4 shows that changing the decoder has very little impact on the detection metrics. Notably, using the diffusion-based decoder reduces the AUC only by around 0.01.

10.4.2 Model fine-tuning

One potential attack could be to remove the watermark through "model purification", which involves fine-tuning the audio LM on a non-watermarked dataset. To test this we fine-tune with different learning rates the small version of the model using a different internal music dataset D_{FT} of similar size without watermarks. For each setting, the model is trained for 20 epochs (10% of the total pre-training steps). We then generate 10k samples with the purified models and obtain scores through the watermark detector.

For each experiment, we report the accuracy obtained for the best threshold on the detection score as well as the TPR when the threshold is chosen such that the FPR is at 10^{-3}. We also report the Fréchet Audio Distance (FAD) (Kilgour et al., 2019) that evaluates the quality of the generative model. We include as a reference the performance of the model before fine-tuning and of a model trained from scratch on D_{FT}.

Table 10.5 suggests that a higher learning rate during fine-tuning makes watermarks more difficult to detect, but it also causes the distribution of generated data to deviate further from the protected model's dataset; at larger learning rates, the distribution has a similar FAD to a model trained from scratch on different data. In other words, since the FAD is almost the same as a model trained from scratch, it may not be worthwhile to start from the watermarked model.

Table 10.5 – Performances of the model after fine-tuning for 20 epochs with different learning rates on a non watermarked dataset. No FT is the model before fine-tuning, scratch is a model trained from scratch on the non watermaked dataset.

LR	Best Acc.	TPR @ FPR=10^{-3}	FAD (↓)
No FT	0.993	0.983	2.067
2×10^{-6}	0.964	0.765	2.908
5×10^{-6}	0.928	0.456	3.149
2×10^{-5}	0.833	0.079	3.344
5×10^{-5}	0.721	0.019	3.740
1×10^{-4}	0.721	0.018	3.643
Scratch	N/A	N/A	3.839

10.5 Conclusion & Discussion

This work introduced a straightforward yet effective approach to watermarking audio language models. This is done via watermarking their training data in a way that is robust to the compression algorithm used to create tokens. It does not require modifications to the model architecture or the training process. Our method is the first to watermark at the latent level and is robust to changes in the decoding process. The main drawback of the current approach is that it requires training the model from scratch, which may be difficult for versioning large models or for adapting already-trained models. While there is a slight decrease in robustness to audio edits compared to post-hoc watermarking, this method allows for keeping the watermark in situations for which post-hoc watermarking is not suitable (open-sourcing...). In conclusion, our method can help trace content origin for open sourced models. It is not a standalone solution and should be complemented with measures like policies, education or monitoring.

Chapter 11

Conclusion

This thesis has focused on advancing the responsible development of generative audio systems based on discrete representations, structured around three key stages: tokenization, modeling, and decoding. Within this framework, we have made several core contributions spanning generation quality, controllability, and content provenance.

On the modeling side, we introduced methods that enhance both the decoding and controllability components of the generative pipeline. Multi-Band Diffusion provides a high-fidelity alternative to standard vocoders, producing more natural results across speech, music, and environmental sounds. MusicGen-Stem extends controllability to the musical domain by enabling multi-stem generation and precise, iterative stem-level editing. Together, these approaches exemplify how architectural choices can make generative systems not only more realistic but also more controllable and user-aligned. They demonstrate a shift from black-box, all-or-nothing generation toward a structured, human-centered paradigm of audio synthesis, where creative intent and model behavior are more closely aligned.

On the safety side, we addressed the verification of content provenance using two complementary watermarking approaches: a post-hoc waveform solution and an in-model solution embedded in the audio LM weights. Localized watermarking of Audioseal achieves state-of-the-art robustness and imperceptibility. Building on this, by ensuring waveform-based watermarking was robust to audio codec tokenization, we demonstrated that in-

model watermarking is feasible by training the model on watermarked data. This approach achieves strong detection performance even when using different NAC decoders or after fine-tuning the LM. These results show that watermarking can be integrated as a core, pipeline-aware component without degrading generation quality, demonstrating that open-source models can be both high-performing and responsibly deployed.

By tackling generation quality, controllability, and content provenance within a unified framework, this work shows that trustworthy generative audio requires a holistic, end-to-end approach. Progress in encoding, modeling, and decoding must evolve alongside safeguards that preserve openness while mitigating misuse. In this way, we can continue advancing generative audio research responsibly, keeping innovation and safety in balance. While scaling models and data can naturally improve performance, certain characteristics of current systems reveal deeper opportunities for progress. One critical area concerns neural audio codecs, which compress waveforms by directly approximating their raw signals. As a result, perceptually identical sounds such as a waveform x and its inversion $-x$ may occupy distant regions in the latent space. Designing latent spaces invariant to perceptually equivalent signals, while maintaining compression efficiency, remains an open challenge. Solving this problem could enable the development of perceptually grounded quantized representations that bridge the gap between acoustic and semantic audio tokens. Such tokens may enhance compression efficiency, improve latent interpretability, and simplify the training of associated language models.

Those autoregressive models operating on discrete representations also face fidelity losses due to the quantization steps inherent in residual vector quantization. Although increasing the number of codebooks might appear to solve this issue, it makes learning meaningful representations exponentially more difficult as later codebooks tend to capture only residual noise. This limitation could be mitigated by approaches that explicitly model dependencies between quantization stages, such as Qinco (Huijben et al., 2024; Vallaeys et al., 2025), though those methods still struggle when trained jointly with autoencoders. An alternative direction involves replacing discrete quantization altogether with a diffusion-like objective in the latent space, as explored by Rouard et al. (2025). Such approaches may offer the advantages of both autoregressive and diffusion-based models, potentially surpassing standard token-based LMs in both quality and generation speed. Addressing these codec limitations could help with the scalability of audio language models. Future research that merges perceptual signal theory, information theory,

and self-supervised representation learning could yield codecs that not only compress efficiently but also encode meaningful auditory abstractions, effectively turning the latent space itself into a semantically rich medium for creative manipulation.

Regarding controllability, significant challenges remain before generative systems can meet the standards of professional music production. Most existing models rely on text prompts as their primary control modality, yet natural language offers only a coarse description of musical content. Text-based conditioning struggles to capture the nuances that define music in practice. Elements such as rhythm, harmonic progression, instrumentation, or timbral balance are difficult to express linguistically and even harder to map consistently to audio representations. Another path toward controllable music generation lies in integrating structured musical representations, such as symbolic annotations, stems, or time-aligned captions, directly into training and inference pipelines. These intermediate representations provide explicit handles over composition and mixing decisions while remaining compatible with existing production workflows. Achieving this will require datasets that couple audio with fine-grained musical metadata, ideally curated or verified by expert musicians to ensure semantic accuracy. Developing scalable methods for collecting and aligning such annotations, similar in spirit to the iterative human-in-the-loop labeling used by Segment Anything (Ravi et al., 2024), could substantially narrow the gap between model control and professional creative needs. In the long term, controllable audio models should aim not merely to generate plausible sounds but to act as transparent and precise tools that respond predictably to user intent within established music-production paradigms.

Nevertheless, important challenges remain. As discussed in Chapter 9, distributing watermark generators and detectors securely is still an open problem. This question is not merely technical; it lies at the heart of trust and transparency in the generative audio ecosystem. If detectors remain proprietary or closed, users and institutions have no independent way to verify whether a piece of audio was generated by a model or originated from a real recording. This lack of transparency undermines the credibility of watermarking as a mechanism for accountability. On the other hand, releasing detectors publicly, or under verifiable and auditable conditions, introduces its own risks, since a fully open detector could allow malicious users to identify and remove embedded marks. It is therefore crucial to find a balance between openness and robustness. Approaches such as zero-knowledge verification protocols (Adelsbach and Sadeghi, 2001; Goldwasser et al.,

1989), cryptographic commitments (Pedersen, 1991), or federated validation frameworks could provide solutions. Developing mechanisms that enable public verifiability while preventing misuse would make watermarking a practical and trustworthy tool for creators, researchers, and policymakers, ensuring that provenance remains auditable across diverse contexts of use.

Another promising avenue concerns watermarking for attribution, which extends beyond provenance to questions of ownership and value distribution. Attribution watermarking could help ensure that, when a user decides to remove their data from a source, it is not inadvertently used to generate model outputs. Such a mechanism supports compliance with evolving data protection and copyright regulations. However, its potential goes further: reliable attribution systems could form the basis for reward mechanisms that fairly recognize the human creators whose works contribute to generative model training. By linking generated content back to the original creators or datasets, either probabilistically or deterministically, it becomes conceivable to design compensation frameworks similar to royalties in music. Such systems would make generative AI ecosystems more sustainable and ethically grounded, promoting cooperation rather than conflict between human artists and AI models. Exploring watermarking as a bridge between technical traceability and creative accountability thus represents an exciting interdisciplinary challenge at the intersection of audio engineering, law, and digital ethics.

The capabilities of generative audio models will continue to improve: outputs will become more realistic, longer, and more structured while offering finer control. Yet, maintaining a parallel focus on safety and responsible deployment will be vital.

This thesis provides a foundation for a more responsible and capable ecosystem of generative audio models—one that can be safely deployed in real-world applications while continuing to push the frontiers of what is possible with token-based audio generation. Ultimately, responsible generative audio envisions a future where human and AI creativity can coexist and collaborate. This thesis represents a step toward that vision, advancing models that are more transparent, controllable, and accountable. These foundations help ensure that the contributions of both humans and AI can be recognized and fairly rewarded. In this way, AI models can act as a partner in the creative process, supporting human expression and enabling new possibilities, while human creators continue to be valued and acknowledged.

References

André Adelsbach and Ahmad-Reza Sadeghi. Zero-knowledge watermark detection and proof of ownership. In *Information Hiding*, 2001. URL https://api.semanticscholar.org/CorpusID:1203115. 139

Andrea Agostinelli, Timo I Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023. 59, 62, 75, 83, 127, 132

Relja Arandjelović and Andrew Zisserman. Look, listen and learn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017a. 33

Relja Arandjelović and Andrew Zisserman. Look, listen and learn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017b. 33

Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus. In *Proceedings of the Twelfth Language Resources and Evaluation Conference (LREC)*, 2020. 70

Sercan Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. Neural voice cloning with a few samples. *Advances in Neural Information Processing Systems*, 31, 2018. 99

Alexei Baevski, Steffen Schneider, and Michael Auli. vq-wav2vec: Self-supervised learning of discrete speech representations. In *International Conference on Learning Representations (ICLR)*, 2020a. 35

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12449–12460. Curran Associates, Inc., 2020b. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf. 35

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, 2015. 59

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. V-JEPA: Latent video prediction for visual representation learning, 2024. URL https://openreview.net/forum?id=WFYbBOEOtv. 31

Mauro Barni and Franco Bartolini. *Watermarking Systems Engineering: Enabling Digital Assets Security and Other Applications*. CRC Press, 2004. 96

Youssef Bassil and Mohammad Alwani. Context-sensitive spelling correction using google web 1t 5-gram information. *Computer and Information Science*, 5(3), April 2012. ISSN 1913-8989. doi: 10.5539/cis.v5n3p37. URL http://dx.doi.org/10.5539/cis.v5n3p37. 47

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 51

Xiaoyu Bie, Xubo Liu, and Gaël Richard. Learning source disentanglement in neural audio codec. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025. 37

Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 10 2016. doi: 10.1145/2964284.2973795. 87

Dmitry Bogdanov, Minz Won, Philip Tovstogan, Alastair Porter, and Xavier Serra. The mtg-jamendo dataset for automatic music tagging. In *Machine Learning for Music Discovery Workshop, International Conference on Machine Learning (ICML 2019)*. ICML, 2019. 70

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matthew Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. Audiolm: A language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2523–2533, 2022. 100

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. Audiolm: a language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023. 57, 127

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *International Conference on Machine Learning (ICML)*, 2012. 41

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020a. 31

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal,

Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020b. 49, 91

Edresson Casanova, Julian Weber, Christopher Shulby, Arnaldo Candido Junior, Eren Gölge, and Moacir Antonelli Ponti. Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone, 2023. URL https://arxiv.org/abs/2112.02418. 99

Yunkee Chae, Woosung Choi, Yuhta Takida, Junghyun Koo, Yukara Ikemiya, Zhi Zhong, Kin Wai Cheuk, Marco A. Martínez-Ramírez, Kyogu Lee, Wei-Hsiang Liao, and Yuki Mitsufuji. Variable bitrate residual vector quantization for audio coding, 2025. URL https://arxiv.org/abs/2410.06016. 53

Guangyu Chen, Yu Wu, Shujie Liu, Tao Liu, Xiaoyong Du, and Furu Wei. Wavmark: Watermarking for audio generation, 2024a. URL https://arxiv.org/abs/2308.12770. 98, 100, 101, 108, 118, 128

Ke Chen, Yusong Wu, Haohe Liu, Marianna Nezhurina, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Musicldm: Enhancing novelty in text-to-music generation using beat-synchronous mixup strategies. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1206–1210. IEEE, 2024b. 60

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations (ICLR)*, 2021a. 40, 67, 68

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, Jian Wu, Long Zhou, Shuo Ren, Yanmin Qian, Yao Qian, Micheal Zeng, and Furu Wei. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16: 1505–1518, 2021b. URL https://api.semanticscholar.org/CorpusID:239885872. 36, 53

Sanyuan Chen, Yu Wu, Chengyi Wang, Shujie Liu, Daniel Tompkins, Zhuo Chen, and Furu Wei. Beats: Audio pre-training with acoustic tokenizers. In *International Conference on Machine Learning (ICML)*, 2023. 33

Sanyuan Chen, Shujie Liu, Long Zhou, Yanqing Liu, Xu Tan, Jinyu Li, Sheng Zhao, Yao Qian, and Furu Wei. Vall-e 2: Neural codec language models are human parity zero-shot text to speech synthesizers, 2024c. URL https://arxiv.org/abs/2406.05370. 57

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework

for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, 2020. 31

Michael Chinen, Felicia SC Lim, Jan Skoglund, Nikita Gureev, Feargus O'Gorman, and Andrew Hines. Visqol v3: An open source production ready objective speech and audio metric. In *International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6. IEEE, 2020. 43, 70, 107

European Commission. European ai act, 2023. URL https://artificialintelligenceact.eu/. Accessed on August 29, 2023. 99

Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. In *Advances in Neural Information Processing Systems*, 2024. 55, 57, 59, 75, 80, 81, 82, 85, 87, 111, 121, 127, 129, 130, 131, 132

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006. ISBN 0471241954. 48

I.J. Cox, J. Kilian, F.T. Leighton, and T. Shamoon. Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing*, 6(12):1673–1687, 1997. doi: 10.1109/83.650120. 96

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su,

Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948. 49

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *Transactions on Machine Learning Research*, 2023. 50, 52, 61, 62, 64, 69, 72, 84, 105, 125, 129, 130

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019a. 36

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019b. 31

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 40

Chris Donahue, Antoine Caillon, Adam Roberts, Ethan Manilow, Philippe Esling, Andrea Agostinelli, Mauro Verzetti, Ian Simon, Olivier Pietquin, Neil Zeghidour, and Jesse Engel. Singsong: Generating musical accompaniments from singing. In *International Conference on Machine Learning (ICML)*, 2023. 81, 83

Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 41

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library, 2025. URL https://arxiv.org/abs/2401.08281. 34

Zhihao Du, Qian Chen, Shiliang Zhang, Kai Hu, Heng Lu, Yexin Yang, Hangrui Hu, Siqi Zheng, Yue Gu, Ziyang Ma, et al. Cosyvoice: A scalable multilingual zero-shot text-to-speech synthesizer based on supervised semantic tokens. *arXiv preprint arXiv:2407.05407*, 2024. 35

Harishchandra Dubey, Vishak Gopal, Ross Cutler, Ashkan Aazami, Sergiy Matusevych, Sebastian Braun, Sefik Emre Eskimez, Manthan Thakker, Takuya Yoshioka, Hannes Gamper, et al. Icassp 2022 deep noise suppression challenge. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 9271–9275. IEEE, 2022. 70

Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. Music source separation in the waveform domain. *IEEE Journal of Selected Topics in Signal Processing*, 2021. 68, 81, 85

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: a speech-text foundation model for real-time dialogue, 2024. URL https://arxiv.org/abs/2410.00037. 52, 53, 56, 58

Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. Clap learning audio concepts from natural language supervision. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023. 33, 59, 87

Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders. In *International Conference on Machine Learning (ICML)*, 2017. 41

Zach Evans, CJ Carr, Josiah Taylor, Scott H. Hawley, and Jordi Pons. Fast timing-conditioned latent audio diffusion. In *International Conference on Machine Learning (ICML)*, 2024. 60

Jianwei Fei, Zhihua Xia, Benedetta Tondi, and Mauro Barni. Supervised gan watermarking for intellectual property protection. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2022. 98

Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 98, 118, 128, 134

Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. Fsd50k: an open dataset of human-labeled sound events. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:829–852, 2021. 70

Cristina Garbacea, Aaron van den Oord, Yazhe Li, Felicia S C Lim, Alejandro Luebs, Oriol Vinyals, and Thomas C Walters. Low bit-rate speech coding with vq-vae and a wavenet decoder. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, page 735–739. IEEE, May 2019. doi: 10.1109/icassp.2019.8683277. URL http://dx.doi.org/10.1109/ICASSP.2019.8683277. 37

Hugo Flores García, Prem Seetharaman, Rithesh Kumar, and Bryan Pardo. Vampnet: Music generation via masked acoustic token modeling. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2023. 59

Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017a. 70

Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017b. 121

Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 33

Brian R Glasberg and Brian C.J Moore. Derivation of auditory filter shapes from notched-noise data. *Hearing Research*, 47(1):103–138, 1990. ISSN 0378-5955. doi: https://doi.org/10.1016/0378-5955(90)90170-T. URL https://www.sciencedirect.com/science/article/pii/0378595590901 70T. 43

Josh A. Goldstein, Girish Sastry, Micah Musser, Renee DiResta, Matthew Gentzel, and Katerina Sedova. Generative language models and automated influence operations: Emerging threats and potential mitigations, 2023. URL https://arxiv.org/abs/2301.04246. 91

Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. In *SIAM journal on computing (Print)*, 1989. URL https://api.semanticscholar.org/CorpusID:8689051. 139

Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer. In *Proc. Interspeech*, 2021. 33

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *Advances in Neural Information Processing Systems*, 2014. 39

Alexey Gritsenko, Tim Salimans, Rianne van den Berg, Jasper Snoek, and Nal Kalchbrenner. A spectral energy distance for parallel speech synthesis. *Advances in Neural Information Processing Systems*, 33:13062–13072, 2020. 105

Yiwei Guo, Zhihan Li, Chenpeng Du, Hankun Wang, Xie Chen, and Kai Yu. Lscodec: Low-bitrate and speaker-decoupled discrete speech codec. *arXiv preprint arXiv:2410.15764*, 2024.

37

Michael Hassid, Tal Remez, Tu Anh Nguyen, Itai Gat, Alexis Conneau, Felix Kreuk, Jade Copet, Alexandre Defossez, Gabriel Synnaeve, Emmanuel Dupoux, et al. Textually pretrained speech language models. *Advances in Neural Information Processing Systems*, 36:63483–63501, 2023. 58

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 31, 40, 64, 65, 91

Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res.*, 23(47):1–33, 2022. 40, 56

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning (ICML)*, 2023. 63, 68

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, PP:1–1, 10 2021. doi: 10.1109/TASLP.2021.3122291. 36, 37, 39, 58, 61

Wei-Ning Hsu, Tal Remez, Bowen Shi, Jacob Donley, and Yossi Adi. Revise: Self-supervised speech resynthesis with visual input for universal and generalized speech enhancement. *arXiv preprint arXiv:2212.11377*, 2022. 62

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. In *International Conference on Learning Representations (ICLR)*, 2019. 41

Huayang Huang, Yu Wu, and Qian Wang. Robin: Robust and invisible watermarks for diffusion models with adversarial optimization. *Advances in Neural Information Processing Systems*, 37:3937–3963, 2024. 98

Qingqing Huang, Aren Jansen, Joonseok Lee, Ravi Ganti, Judith Yue Li, and Daniel PW Ellis. Mulan: A joint embedding of music audio and natural language. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2022a. 33, 59

Qingqing Huang, Daniel S Park, Tao Wang, Timo I Denk, Andy Ly, Nanxin Chen, Zhengdong Zhang, Zhishuai Zhang, Jiahui Yu, Christian Frank, et al. Noise2music: Text-conditioned music generation with diffusion models. *arXiv preprint arXiv:2302.03917*, 2023. 60, 63

Wen-Chin Huang, Shu-Wen Yang, Tomoki Hayashi, Hung-Yi Lee, Shinji Watanabe, and Tomoki Toda. S3prl-vc: Open-source voice conversion framework with self-supervised speech repre-

sentations. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6552–6556. IEEE, 2022b. 62

Iris A. M. Huijben, Matthijs Douze, Matthew Muckley, Ruud J. G. van Sloun, and Jakob Verbeek. Residual quantization with implicit neural codebooks. In *International Conference on Machine Learning (ICML)*, 2024. 138

International Telecommunication Union. Method for the subjective assessment of intermediate quality level of audio systems (MUSHRA). Technical Report ITU-R BS.1534-3, International Telecommunication Union, Radiocommunication Assembly, 2015. https://www.itu.int/rec/R-REC-BS.1534-3-201510-I/en. 45

Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010. 35

Zeqian Ju, Yuancheng Wang, Kai Shen, Xu Tan, Detai Xin, Dongchao Yang, Yanqing Liu, Yichong Leng, Kaitao Song, Siliang Tang, et al. Naturalspeech 3: zero-shot speech synthesis with factorized codec and diffusion models. In *International Conference on Machine Learning (ICML)*, pages 22605–22623, 2024. 37, 60

Lauri Juvela and Xin Wang. Collaborative watermarking for adversarial speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024a. 101, 134

Lauri Juvela and Xin Wang. Collaborative watermarking for adversarial speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11231–11235. IEEE, 2024b. 98

Tornike Karchkhadze, Mohammad Rasool Izadi, Ke Chen, Gerard Assayag, and Shlomo Dubnov. Multi-track musicldm: Towards versatile music generation with latent diffusion model. *arXiv preprint arXiv:2409.02845*, 2024a. 84

Tornike Karchkhadze, Mohammad Rasool Izadi, and Shlomo Dubnov. Simultaneous music separation and generation using multi-track latent diffusion models. *arXiv preprint arXiv:2409.12346*, 2024b. 84

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, 2022. 66

Hasam Khalid, Shahroz Tariq, and Simon S. Woo. Fakeavceleb: A novel audio-video multimodal deepfake dataset. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021. 111, 121, 122

Eugene Kharitonov, Ann Lee, Adam Polyak, Yossi Adi, Jade Copet, Kushal Lakhotia, Tu Anh

Nguyen, Morgane Riviere, Abdelrahman Mohamed, Emmanuel Dupoux, et al. Text-free prosody-aware generative spoken language modeling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8666–8681, 2022a. 62

Eugene Kharitonov, Ann Lee, Adam Polyak, Yossi Adi, Jade Copet, Kushal Lakhotia, Tu-Anh Nguyen, Morgane Rivière, Abdelrahman Mohamed, Emmanuel Dupoux, and Wei-Ning Hsu. Text-free prosody-aware generative spoken language modeling, 2022b. URL https://arxiv.org/abs/2109.03264. 131

Eugene Kharitonov, Damien Vincent, Zalán Borsos, Raphaël Marinier, Sertan Girgin, Olivier Pietquin, Matt Sharifi, Marco Tagliasacchi, and Neil Zeghidour. Speak, read and prompt: High-fidelity text-to-speech with minimal supervision. *Transactions of the Association for Computational Linguistics*, 2023a. 75

Eugene Kharitonov, Damien Vincent, Zalán Borsos, Raphaël Marinier, Sertan Girgin, Olivier Pietquin, Matthew Sharifi, Marco Tagliasacchi, and Neil Zeghidour. Speak, read and prompt: High-fidelity text-to-speech with minimal supervision. *Transactions of the Association for Computational Linguistics*, 2023b. 100

Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A metric for evaluating music enhancement algorithms. In *Proc. Interspeech*, 2019. 44, 87, 135

Changhoon Kim, Kyle Min, Maitreya Patel, Sheng Cheng, and Yezhou Yang. Wouaf: Weight modulation for user attribution and fingerprinting in text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 118, 128

Jaehyeon Kim, Jungil Kong, and Juhee Son. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *International Conference on Machine Learning (ICML)*, 2021. 99

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 30

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International Conference on Machine Learning (ICML)*, pages 17061–17084. PMLR, 2023. 98

Darko Kirovski and Hagai Attias. Audio watermark robustness to desynchronization via beat detection. In *International workshop on information hiding*, pages 160–176. Springer, 2002. 97

Darko Kirovski and Hagai Attias. Audio watermark robustness to desynchronization via beat

detection. In Fabien A. P. Petitcolas, editor, *Information Hiding*, pages 160–176, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-36415-3. 105

Darko Kirovski and Henrique S Malvar. Spread-spectrum watermarking of audio signals. *IEEE transactions on signal processing*, 51(4):1020–1033, 2003. 97

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020. 39, 62, 72, 128, 134

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations (ICLR)*, 2021. 40, 68, 134

Felix Kreuk, Adam Polyak, Jade Copet, Eugene Kharitonov, Tu Anh Nguyen, Morgan Rivière, Wei-Ning Hsu, Abdelrahman Mohamed, Emmanuel Dupoux, and Yossi Adi. Textless speech emotion conversion using discrete & decomposed representations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11200–11214, 2022. 62

Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. Audiogen: Textually guided audio generation. In *International Conference on Learning Representations (ICLR)*, 2023. 57, 62, 111, 121, 127

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf. 30

Kundan Kumar, Rithesh Kumar, Thibault De Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in Neural Information Processing Systems*, 32, 2019. 39, 61, 128, 134

Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved rvqgan. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 27980–27993. Curran Associates, Inc., 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/58d0e78cf042af5876e12661087bea12-Paper-Conference.pdf. 52

Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved rvqgan. In *Advances in Neural Information*

Processing Systems, 2023b. 72

Kushal Lakhotia, Evgeny Kharitonov, Wei-Ning Hsu, Yossi Adi, Adam Polyak, Benjamin Bolte, Tu-Anh Nguyen, Jade Copet, Alexei Baevski, Adelrahman Mohamed, and Emmanuel Dupoux. Generative spoken language modeling from raw audio. *Transactions of the Association for Computational Linguistics*, 2021. 35, 58

Matthew Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, and Wei-Ning Hsu. Voicebox: Text-guided multilingual universal speech generation at scale. In *Advances in Neural Information Processing Systems*, 2024. 60, 91, 100, 109, 126

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4): 541–551, 1989. doi: 10.1162/neco.1989.1.4.541. 30

Sang-gil Lee, Heeseung Kim, Chaehun Shin, Xu Tan, Chang Liu, Qi Meng, Tao Qin, Wei Chen, Sungroh Yoon, and Tie-Yan Liu. Priorgrad: Improving conditional denoising diffusion models with data-driven adaptive prior. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022. 40, 68, 72

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020. 48

Peike Patrick Li, Boyu Chen, Yao Yao, Yikai Wang, Allen Wang, and Alex Wang. Jen-1: Text-guided universal music generation with omnidirectional diffusion models. In *IEEE Conference on Artificial Intelligence (CAI)*, pages 762–769. IEEE, 2024. 60

Wen-Nung Lie and Li-Chun Chang. Robust and high-quality time-domain audio watermarking based on low-frequency amplitude modification. *IEEE transactions on multimedia*, 8(1):46–59, 2006. 96

Andy T Liu, Po-chun Hsu, and Hung-yi Lee. Unsupervised end-to-end learning of discrete linguistic units for voice conversion. *arXiv preprint arXiv:1905.11563*, 2019. 62

Chang Liu, Jie Zhang, Han Fang, Zehua Ma, Weiming Zhang, and Nenghai Yu. Dear: A deep-learning-based audio re-recording resilient watermarking. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 13201–13209. AAAI Press, 2023a. doi: 10.1609/aaai.v37i11.26550. 98, 100

Chang Liu, Jie Zhang, Tianwei Zhang, Xi Yang, Weiming Zhang, and Nenghai Yu. Detecting voice cloning attacks via timbre watermarking. *arXiv preprint arXiv:2312.03410*, 2023b. 98

Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. AudioLDM: Text-to-audio generation with latent diffusion models. In *International Conference on Machine Learning (ICML)*, pages 21450–21474, 2023c. 60, 127

Haohe Liu, Xuenan Xu, Yi Yuan, Mengyue Wu, Wenwu Wang, and Mark D. Plumbley. Semanticodec: An ultra low bitrate semantic audio codec for general sound. *IEEE Journal of Selected Topics in Signal Processing*, 18(8):1448–1461, December 2024a. ISSN 1941-0484. doi: 10.1109/jstsp.2024.3506286. URL http://dx.doi.org/10.1109/JSTSP.2024.3506286. 50, 53

Haohe Liu, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Qiao Tian, Yuping Wang, Wenwu Wang, Yuxuan Wang, and Mark D. Plumbley. Audioldm 2: Learning holistic audio generation with self-supervised pretraining. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:2871–2883, 2024b. doi: 10.1109/TASLP.2024.3399607. 60

Weizhi Liu, Yue Li, Dongdong Lin, Hui Tian, and Haizhou Li. Groot: Generating robust watermark for diffusion-model-based audio synthesis. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 3294–3302, 2024c. 98

Xuechen Liu, Xin Wang, Md Sahidullah, Jose Patino, Héctor Delgado, Tomi Kinnunen, Massimiliano Todisco, Junichi Yamagishi, Nicholas Evans, Andreas Nautsch, et al. Asvspoof 2021: Towards spoofed and deepfake speech detection in the wild. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023d. 111, 121, 122

Yi Luo, Zhuo Chen, and Takuya Yoshioka. Dual-path rnn: efficient long sequence modeling for time-domain single-channel speech separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 46–50. IEEE, 2020. 120

J. Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, 1975. doi: 10.1109/PROC.1975.9792. 28

Ethan Manilow, Gordon Wichern, Prem Seetharaman, and Jonathan Le Roux. Cutting music source separation some slakh: A dataset to study the impact of training data quality and quantity. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019. 88

Hila Manor and Tomer Michaeli. Zero-shot unsupervised and text-based audio editing using ddpm inversion. In *ICML*, 2024. 82

Giorgio Mariani, Irene Tallini, Emilian Postolache, Michele Mancusi, Luca Cosmo, and Emanuele Rodolà. Multi-source diffusion models for simultaneous music generation and separation. In *International Conference on Learning Representations (ICLR)*, 2024. 81, 84, 88

Álvaro Martín-Cortinas, Daniel Sáez-Trigueros, Iván Vallés-Pérez, Biel Tura-Vecino, Piotr Biliński, Mateusz Lajszczak, Grzegorz Beringer, Roberto Barra-Chicote, and Jaime Lorenzo-Trueba. Enhancing the stability of llm-based speech generation systems through self-supervised representations. *arXiv preprint arXiv:2402.03407*, 2024. 37

Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. In *International Conference on Learning Representations (ICLR)*, 2017. 41

Margi Murphy, Rachel Metz, Mark Bergen, and Bloomberg. Biden audio deepfake spurs ai startup elevenlabs—valued at $1.1 billion—to ban account: 'we're going to see a lot more of this'. *Fortune*, January 2024. URL https://fortune.com/2024/01/27/ai-firm-elevenlabs-bans-account-for-biden-audio-deepfake/. 99, 116

Tu Anh Nguyen, Wei-Ning Hsu, Antony D'Avirro, Bowen Shi, Itai Gat, Maryam Fazel-Zarani, Tal Remez, Jade Copet, Gabriel Synnaeve, Michael Hassid, Felix Kreuk, Yossi Adi, and Emmanuel Dupoux. Expresso: A benchmark and analysis of discrete expressive speech resynthesis. In *Proc. Interspeech*, 2023. 111, 121

Tu Anh Nguyen, Benjamin Muller, Bokai Yu, Marta R. Costa-jussa, Maha Elbayad, Sravya Popuri, Paul-Ambroise Duquenne, Robin Algayres, Ruslan Mavlyutov, Itai Gat, Gabriel Synnaeve, Juan Pino, Benoit Sagot, and Emmanuel Dupoux. Spirit-lm: Interleaved spoken and written language model. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. 35, 58

Zhicheng Ni, Yun-Qing Shi, Nirwan Ansari, and Wei Su. Reversible data hiding. *IEEE Transactions on circuits and systems for video technology*, 16(3):354–362, 2006. 96

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning (ICML)*, pages 8162–8171. PMLR, 2021. 67, 74

Nikos Nikolaidis and Ioannis Pitas. Robust image watermarking in the spatial domain. *Signal processing*, 66(3):385–403, 1998. 96

Zachary Novack, Julian McAuley, Taylor Berg-Kirkpatrick, and Nicholas J. Bryan. Ditto: Diffusion inference-time t-optimization for music generation. In *International Conference on Machine Learning (ICML)*, 2024. 83

Cyberspace Administration of China. Chinese ai governance rules, 2023. URL http://www.cac.gov.cn/2023-07/13/c_1690898327029107.htm. Accessed on August 29, 2023. 99

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model

for raw audio. *arXiv preprint arXiv:1609.03499*, 2016. Presented at the 9th ISCA Speech Synthesis Workshop (SSW9). 41, 49, 61

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Rad-

ford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774. 49, 127, 131

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. 31

Patrick O'Reilly, Zeyu Jin, Jiaqi Su, and Bryan Pardo. Maskmark: Robust neuralwatermarking for real and synthetic speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4650–4654. IEEE, 2024. 97

Julian D Parker, Anton Smirnov, Jordi Pons, CJ Carr, Zack Zukowski, Zach Evans, and Xubo Liu. Scaling transformers for low-bitrate high-quality speech coding, 2024a. URL https://arxiv.org/abs/2411.19842. 52

Julian D Parker, Anton Smirnov, Jordi Pons, CJ Carr, Zack Zukowski, Zach Evans, and Xubo Liu. Scaling transformers for low-bitrate high-quality speech coding, 2024b. URL https://arxiv.org/abs/2411.19842. 52

Julian D. Parker, Janne Spijkervet, Katerina Kosta, Furkan Yesiler, Boris Kuznetsov, Ju-Chiang Wang, Matt Avent, Jitong Chen, and Duc Le. Stemgen: A music generation model that listens. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024c. 81, 83

Santiago Pascual, Gautam Bhattacharya, Chunghsin Yeh, Jordi Pons, and Joan Serrà. Full-band general audio synthesis with score-based diffusion, 2022. 40

Kosta Pavlović, Slavko Kovačević, Igor Djurović, and Adam Wojciechowski. Robust speech watermarking by a jointly trained embedder and detector using a dnn. *Digital Signal Processing*, 122:103381, 2022. 97, 100

Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual International Cryptology Conference*, 1991. URL https://api.semanticscholar.org/CorpusID:13447431. 140

Vincent Persichetti. *Twentieth-Century Harmony*. W. W. Norton & Company, 1961. 87

Manos Plitsis, Theodoros Kouzelis, Georgios Paraskevopoulos, Vassilis Katsouros, and Yannis Panagakis. Investigating personalization methods in text to music generation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024. 83

Adam Polyak, Yossi Adi, Jade Copet, Eugene Kharitonov, Kushal Lakhotia, Wei-Ning Hsu, Abdelrahman Mohamed, and Emmanuel Dupoux. Speech resynthesis from discrete disentangled self-supervised representations. In *Proc. Interspeech*, 2021a. 37

Adam Polyak, Yossi Adi, Jade Copet, Eugene Kharitonov, Kushal Lakhotia, Wei-Ning Hsu, Abdelrahman Mohamed, and Emmanuel Dupoux. Speech resynthesis from discrete disentangled self-supervised representations. In *Proc. Interspeech*, 2021b. 39, 62

Lawrence R. Rabiner and Ronald W. Schafer. *Digital Processing of Speech Signals*. 1978. URL https://api.semanticscholar.org/CorpusID:264163851. 27

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PmLR, 2021. 33, 59

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning (ICML)*, 2023. 32

Colin Raffel, Brian Mcfee, Eric Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel Ellis. mir_eval: A transparent implementation of common mir metrics. In *Proceedings - 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 10 2014. 87

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 40

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and

Christoph Feichtenhofer. Sam 2: Segment anything in images and videos, 2024. URL https://arxiv.org/abs/2408.00714. 139

Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech. In *Advances in Neural Information Processing Systems*, 2019. 41

Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech. In *International Conference on Learning Representations (ICLR)*, 2021. 41

Alain Riou, Stefan Lattner, Gaëtan Hadjeres, and Geoffroy Peeters. Pesto: Pitch estimation with self-supervised transposition-equivariant objective. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. International Society for Music Information Retrieval, 2023. 87

Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 749–752. IEEE, 2001. 107

Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In *International Conference on Machine Learning (ICML)*, 2018. 41

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 127

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015. 50, 68

Simon Rouard, Yossi Adi, Jade Copet, Axel Roebel, and Alexandre Défossez. Audio conditioning for music generation via discrete bottleneck features, 2024. URL https://arxiv.org/abs/2407.12563. 59, 82, 83

Simon Rouard, Manu Orsini, Axel Roebel, Neil Zeghidour, and Alexandre Défossez. Continuous audio language models, 2025. URL https://arxiv.org/abs/2509.06926. 138

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024. URL https://arxiv.org/abs/2308.12950. 49

Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images, 2019. URL https://arxiv.org/abs/1901.08971. 92

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. 40, 91

Robin San Roman, Yossi Adi, Antoine Deleforge, Romain Serizel, Gabriel Synnaeve, and Alexandre Defossez. From discrete tokens to high-fidelity audio using multi-band diffusion. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 1526–1538. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/054f771d614df12fe8def8ecdbe4e8e1-Paper-Conference.pdf. 61, 128, 134

Robin San Roman, Pierre Fernandez, Alexandre Défossez, Teddy Furon, Tuan Tran, and Hady Elsahar. Proactive detection of voice cloning with localized watermarking. In *International Conference on Machine Learning (ICML)*, 2024. 99, 127, 128, 129, 130, 131

Robin San Roman, Pierre Fernandez, Antoine Deleforge, Yossi Adi, and Romain Serizel. Latent watermarking of audio generative models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025. 127

Khalid Sayood. *Introduction to Data Compression*. 1996. URL https://api.semanticscholar.org/CorpusID:36835250. 28

Jan Schnupp, Israel Nelken, and Andrew King. *Auditory neuroscience: Making sense of sound*. MIT press, 2011. 25, 43, 65, 68, 105

Seamless Communication, Loïc Barrault, Yu-An Chung, Mariano Coria Meglioli, David Dale, Ning Dong, Mark Duppenthaler, Paul-Ambroise Duquenne, Brian Ellis, Hady Elsahar, Justin Haaheim, John Hoffman, Min-Jae Hwang, Hirofumi Inaguma, Christopher Klaiber, Ilia Kulikov, Pengwei Li, Daniel Licht, Jean Maillard, Ruslan Mavlyutov, Alice Rakotoarison, Kaushik Ram Sadagopan, Abinesh Ramakrishnan, Tuan Tran, Guillaume Wenzek, Yilin Yang, Ethan Ye, Ivan Evtimov, Pierre Fernandez, Cynthia Gao, Prangthip Hansanti, Elahe Kalbassi, Amanda Kallet, Artyom Kozhevnikov, Gabriel Mejia, Robin San Roman, Christophe Touret, Corinne Wong, Carleigh Wood, Bokai Yu, Pierre Andrews, Can Balioglu, Peng-Jen Chen, Marta R. Costa-jussà, Maha Elbayad, Hongyu Gong, Francisco Guzmán, Kevin Heffernan, Somya Jain, Justine Kao, Ann Lee, Xutai Ma, Alex Mourachko, Benjamin Peloquin, Juan Pino, Sravya Popuri, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, Anna Sun, Paden Tomasello, Changhan Wang, Jeff Wang, Skyler Wang, and Mary Williamson. Seamless: Multilingual expressive and streaming speech translation. 2023. 111, 121, 128

B Series. Method for the subjective assessment of intermediate quality level of audio systems. *International Telecommunication Union Radiocommunication Assembly*, 2014a. 108

B Series. Method for the subjective assessment of intermediate quality level of audio systems. *International Telecommunication Union Radiocommunication Assembly*, 2014b. 70

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x. 24

Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018. 41

Kai Shen, Zeqian Ju, Xu Tan, Yanqing Liu, Yichong Leng, Lei He, Tao Qin, Sheng Zhao, and Jiang Bian. Naturalspeech 2: Latent diffusion models are natural and zero-shot speech and singing synthesizers. In *International Conference on Learning Representations (ICLR)*, 2024. 60

Hubert Siuzdak, Florian Grötschla, and Luca A Lanzendörfer. Snac: Multi-scale neural audio codec. In *Audio Imagination: NeurIPS 2024 Workshop AI-Driven Speech, Music, and Sound Generation*, 2024. 53

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021. 68

Andreas Spanias, Ted Painter, and Venkatraman Atti. *Audio Signal Processing and Coding*. 2007. URL https://api.semanticscholar.org/CorpusID:62755817. 26

Giovanni Spitale, Nikola Biller-Andorno, and Federico Germani. Ai model gpt-3 (dis) informs us better than humans. *Science Advances*, 9(26):eadh1850, 2023. 91

Nick Statt. Satanic figures are hidden in doom's soundtrack, 2016. URL https://www.theverge.com/2016/5/31/11825606/doom-2016-soundtrack-satan-666-inverted-pentagram. Accessed: 2025-10-12. 92

Suno. Suno—ai for music creators. https://www.suno.com, 2025. 59

Cees H Taal, Richard C Hendriks, Richard Heusdens, and Jesper Jensen. A short-time objective intelligibility measure for time-frequency weighted noisy speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4214–4217. IEEE, 2010. 108

Naoya Takahashi, Mayank Kumar, Yuki Mitsufuji, et al. Hierarchical diffusion models for singing voice neural vocoder. In *IEEE International Conference on Acoustics, Speech and*

Signal Processing (ICASSP), pages 1–5. IEEE, 2023. 40, 63

Or Tal, Alon Ziv, Itai Gat, Felix Kreuk, and Yossi Adi. Joint audio and symbolic conditioning for temporally controlled text-to-music generation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2024. 80

Xu Tan, Tao Qin, Frank Soong, and Tie-Yan Liu. A survey on neural speech synthesis. *arXiv preprint arXiv:2106.15561*, 2021. 61

International telecommunication Union. Algorithms to measure audio programme loudness and true-peak audio level. *Series, BS*, 2011. 105

Keiichi Tokuda, Heiga Zen, and Alan W. Black. An hmm-based speech synthesis system applied to english. In *Proceedings of 2002 IEEE Workshop on Speech Synthesis, 2002.*, pages 227–230, 2002. doi: 10.1109/WSS.2002.1224415. 41

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971. 49, 91, 131

Udio. Udio: Ai music made simple. https://www.udio.com, 2025. 59

USA. Ensuring safe, secure, and trustworthy ai. https://www.whitehouse.gov/wp-content/uploads/2023/07/Ensuring-Safe-Secure-and-Trustworthy-AI.pdf, July 2023. Accessed: [july 2023]. 99

Jean-Marc Valin, Koen Vos, and Timothy Terriberry. Definition of the opus audio codec. Technical report, RFC Editor, 2012. 62, 71

Théophane Vallaeys, Matthew Muckley, Jakob Verbeek, and Matthijs Douze. Qinco2: Vector compression and search with improved implicit neural codebooks, 2025. URL https://arxiv.org/abs/2501.03078. 138

Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf. 51

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. 34, 37, 40, 61

Benjamin van Niekerk, Leanne Nortje, and Herman Kamper. Vector-quantized neural networks for acoustic unit discovery in the zerospeech 2020 challenge. *arXiv preprint arXiv:2005.09409*, 2020. 39

Ron G Van Schyndel, Andrew Z Tirkel, and Charles F Osborne. A digital watermark. In

Proceedings of 1st international conference on image processing, volume 2, pages 86–90. IEEE, 1994. 96

A Vasuki and PT Vanathi. A review of vector quantization techniques. *IEEE Potentials*, 25(4): 39–47, 2006. 50

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 48, 54

Apoorv Vyas, Bowen Shi, Matthew Le, Andros Tjandra, Yi-Chiao Wu, Baishan Guo, Jiemin Zhang, Xinyue Zhang, Robert Adkins, William Ngan, Jeff Wang, Ivan Cruz, Bapi Akula, Akinniyi Akinyemi, Brian Ellis, Rashel Moritz, Yael Yungster, Alice Rakotoarison, Liang Tan, Chris Summers, Carleigh Wood, Joshua Lane, Mary Williamson, and Wei-Ning Hsu. Audiobox: Unified audio generation with natural language prompts, 2023. URL https://arxiv.org/abs/2312.15821. 60, 91, 127, 128

Changhan Wang, Morgane Rivière, Ann Lee, Anne Wu, Chaitanya Talnikar, Daniel Haziza, Mary Williamson, Juan Miguel Pino, and Emmanuel Dupoux. Voxpopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 993–1003. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.ACL-LONG.80. URL https://doi.org/10.18653/v1/2021.acl-long.80. 107

Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, et al. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*, 2023a. 57, 62, 75, 79, 91, 99, 127

Yulong Wang, Tong Sun, Shenghong Li, Xin Yuan, Wei Ni, Ekram Hossain, and H. Vincent Poor. Adversarial attacks and defenses in machine learning-powered networks: A contemporary survey, 2023b. URL https://arxiv.org/abs/2303.06302. 94

Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis. In *Proc. Interspeech*, 2017. 41

Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. In *Advances in Neural Information*

Processing Systems, 2023. 98

Hanzhou Wu, Gen Liu, Yuwei Yao, and Xinpeng Zhang. Watermarking neural networks with watermarked images. *IEEE Transactions on Circuits and Systems for Video Technology*, 31 (7):2591–2601, 2020. 98

Shih-Lun Wu, Chris Donahue, Shinji Watanabe, and Nicholas J Bryan. Music controlnet: Multiple time-varying controls for music generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:2692–2703, 2024a. 82

Shih-Lun Wu, Chris Donahue, Shinji Watanabe, and Nicholas J Bryan. Music controlnet: Multiple time-varying controls for music generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:2692–2703, 2024b. 80

Yong Xiang, Iynkaran Natgunanathan, Song Guo, Wanlei Zhou, and Saeid Nahavandi. Patchwork-based audio watermarking method robust to de-synchronization attacks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(9):1413–1423, 2014. 97

Xiph.Org Foundation. FLAC - Free Lossless Audio Codec. `https://xiph.org/flac/`, 2001. Accessed: 2025-06-02. 28

Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203. IEEE, 2020. 52

Yao-Yuan Yang, Moto Hira, Zhaoheng Ni, Anjali Chourdia, Artyom Astafurov, Caroline Chen, Ching-Feng Yeh, Christian Puhrsch, David Pollack, Dmitriy Genzel, Donny Greenberg, Edward Z. Yang, Jason Lian, Jay Mahadeokar, Jeff Hwang, Ji Chen, Peter Goldsborough, Prabhat Roy, Sean Narenthiran, Shinji Watanabe, Soumith Chintala, Vincent Quenneville-Bélair, and Yangyang Shi. Torchaudio: Building blocks for audio and speech processing. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022. 123

Yao Yao, Peike Li, Boyu Chen, and Alex Wang. Jen-1 composer: A unified framework for high-fidelity multi-track music generation. *ArXiv*, 2023a. 81

Yao Yao, Peike Li, Boyu Chen, and Alex Wang. Jen-1 composer: A unified framework for high-fidelity multi-track music generation. *ArXiv*, 2023b. 84

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*, 2019. 104

Ning Yu, Vladislav Skripniuk, Sahar Abdelnabi, and Mario Fritz. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data. In *Proceedings of the*

IEEE/CVF International Conference on Computer Vision (ICCV), pages 14448–14457, 2021. 98, 118

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021. 52, 62, 64, 131

Heiga Zen and Andrew Senior. Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3844–3848, 2014. doi: 10.1109/ICASSP.2014.6854321. 41

Heiga Zen, Andrew Senior, and Mike Schuster. Statistical parametric speech synthesis using deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7962–7966, 2013. doi: 10.1109/ICASSP.2013.6639215. 41

Yixiao Zhang, Yukara Ikemiya, Woosung Choi, Naoki Murata, Marco A. Martínez-Ramírez, Liwei Lin, Gus Xia, Wei-Hsiang Liao, Yuki Mitsufuji, and Simon Dixon. Instruct-musicgen: Unlocking text-to-music editing for music language models via instruction tuning, 2024a. URL https://arxiv.org/abs/2405.18386. 83

Yixiao Zhang, Yukara Ikemiya, Gus Xia, Naoki Murata, Marco A. Martínez-Ramírez, Wei-Hsiang Liao, Yuki Mitsufuji, and Simon Dixon. Musicmagus: Zero-shot text-to-music editing via diffusion models, 2024b. URL https://arxiv.org/abs/2402.06178. 83

Ziqiang Zhang, Long Zhou, Chengyi Wang, Sanyuan Chen, Yu Wu, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. Speak foreign languages with your own voice: Cross-lingual neural codec language modeling, 2023. URL https://arxiv.org/abs/2303.03926. 57

Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 657–672, 2018. 97

Alon Ziv, Itai Gat, Gael Le Lan, Tal Remez, Felix Kreuk, Jade Copet, Alexandre Défossez, Gabriel Synnaeve, and Yossi Adi. Masked audio generation using a single non-autoregressive transformer. In *International Conference on Learning Representations (ICLR)*, 2024. 59