Audio Restoration for Generative Models — Improving MusicGen Outputs

August 29, 2025

Sofia De Angelis

Abstract

Il progetto si focalizza sul miglioramento della qualità percettiva del suono generato da MusicGen mediante un sistema di post-elaborazione capace di ridurre rumori e artefatti, preservando al contempo l'integrità e il contenuto musicale.

1. Introduzione

Prima di illustrare l'approccio adottato nel progetto, è utile descrivere il punto di partenza: **MusicGen**. Si tratta di un modello autoregressivo sviluppato da Meta AI, progettato per trasformare una descrizione testuale (prompt) in un contenuto audio coerente. Il modello è stato addestrato su un ampio dataset di oltre 20.000 ore di musica, che gli permette di riconoscere e riprodurre strutture musicali complesse. Dal punto di vista tecnico, MusicGen elabora i dati audio scomponendoli in unità più semplici, così da facilitare la generazione di nuove sequenze sonore capaci di rispettare lo stile e le caratteristiche indicate nel prompt originario.

La generazione audio tramite modelli di intelligenza artificiale è ancora meno diffusa rispetto ad altri ambiti, poiché rappresenta un campo complesso e relativamente poco esplorato. Pur essendo MusicGen un modello avanzato, le sue prestazioni vengono tuttora valutate soprattutto attraverso analisi di tipo empirico, che includono sia metriche automatiche sia giudizi umani.

Obiettivo. Nel progetto sono partita dall'utilizzo di un modello pre-addestrato, MusicGen, con l'obiettivo di migliorarne l'output attraverso alcune manipolazioni audio effettuate con librerie specifiche e con l'impiego di modelli supplementari, tra cui Demucs. Quest'ultimo è un modello sviluppato da Meta AI per la source separation, ovvero la separazione di un brano musicale nelle sue componenti fondamentali (ad esempio voce, batteria, basso e strumenti rimanenti). Questo processo permette di ridurre rumori, artefatti e imperfezioni presenti nell'audio originale, restituendo un segnale più pulito e tecnicamente più definito.

Lo scopo principale era quello di ottenere un miglioramento qualitativo rispetto all'audio generato direttamente da MusicGen. Pur non trattandosi di un perfezionamento ottimale, i test effettuati hanno mostrato un incremento, seppur minimo, della qualità percepita, confermando la validità dell'approccio e aprendo la strada a possibili sviluppi futuri.

Codice. Il link del codice progetto è : https://github.com/sofiadgelis/ML-AudioEnhancement-Project/blob/d60a28c7b2109251536cfd4a5a14591149c63ceb/project_code.ipynb

2. Modello MusicGen

Andiamo a descrivere il modello di MusicGen (*figure 1*) che si basa su tre componenti principali:

- EnCodec (audio encoder e decoder): comprime i segnali audio in rappresentazioni latenti (token) e consente di ricostruirli successivamente in forma di waveform.
- Trasformatore multi-layer: modella le sequenze di codici latenti catturando relazioni musicali di alto livello e rappresentazioni complesse, permettendo una coerenza globale nel brano generato.
- T5 text encoder: traduce il prompt testuale in embedding numerici che guidano la generazione dell'audio.

Dal punto di vista matematico, EnCodec utilizza la **Residual Vector Quantization (RVQ)**, va a suddividere il processo di quantizzazione su più livelli, ognuno dei quali si occupa dell'errore residuo del precedente. Ciò consente di scalare il sistema per funzionare su diversi bitrate (scalando il numero di strati), quindi va a comprimere l'audio in token mediante più codebook.

Email: Sofia De Angelis <deangelis.2126369@@studenti.uniroma1.it>.

Machine Learning 2025, Sapienza University of Rome, 2nd semester a.y. 2024/2025.

Avendo un audio di riferimento $X \in \mathbb{R}^{d \cdot f_n}$ in cui: d è la durata del segnale e f_n la lunghezza del campionamento, il modello genera una sequenza di codici latenti $Q \in \{1, ..., L\}^{N \times d \cdot f_s}$. In questa formula L è la dimensione del codebook (numero di possibili simboli),N è il numero di codebook impiegati e f_s la frequenza di campionamento nello spazio latente; volendo essere specifici in MusicGen questi valori corrispondono a: $N = 4,\ f_n = 50s\ f_s = 32000\ e\ L = 2048$.
Infine, il trasformatore apprende le relazioni sequenziali tra i codici latenti e permette la generazione di nuove sequenze audio coerenti con il prompt di input.

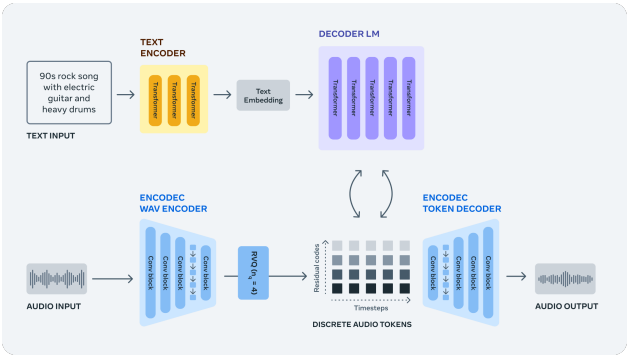

Figure 1. MusicGen overview

3. Approccio e tecniche applicate

Il progetto è stato sviluppato con l'obiettivo di migliorare la qualità dell'audio generato dal modello MusicGen, integrando un flusso di post–processing basato su tecniche di elaborazione del segnale e modelli di separazione audio. Possiamo andare a fare una suddivisione in sezioni:

- Generazione dell'audio: ciò avviene tramite il modello pre-addestrato **MusicGenForConditionalGeneration**, disponibile tramite la libreria Transformers. A partire da un prompt testuale, il modello produce un segnale audio in formato tensoriale, successivamente convertito in file WAV per l'elaborazione.

- Pre–processing e gestione dei file: l'audio iniziale viene caricato e normalizzato tramite le librerie, con lo scopo di uniformare il livello del segnale e ridurre fenomeni di clipping. In questa fase i file vengono inoltre organizzati in directory dedicate per una gestione strutturata degli output.

- Miglioramento con Demucs: per incrementare la qualità percepita del segnale si è fatto ricorso a **Demucs**, un modello neurale originariamente progettato per la separazione delle sorgenti audio. In questo contesto, esso è stato utilizzato come strumento di "audio enhancement", mirato alla riduzione di rumori e artefatti generati in fase di sintesi da MusicGen.

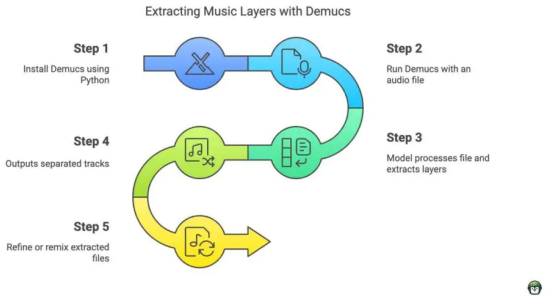

Figure 2. Schema a blocchi del modello Demucs per la separazione delle sorgenti audio

- Post–processing e visualizzazione: gli output finali sono stati ulteriormente elaborati e salvati in cartelle dedicate , con la possibilità di visualizzare spettri e forme d'onda tramite Matplotlib e Librosa.display, al fine di valutare qualitativamente gli effetti del miglioramento.

4. Risultati e conclusione

L'approccio descritto, pur non ottimale, garantisce un funzionamento minimo e coerente. L'audio prodotto da MusicGen risulta semanticamente fedele al prompt, ma spesso affetto da artefatti e instabilità timbriche. L'applicazione di Demucs consente di ridurre tali imperfezioni, anche se con efficacia variabile, poiché il modello non è stato addestrato specificamente per segnali generati artificialmente.

In conclusione, il progetto può essere considerato un proof of concept di pipeline per la generazione e il miglioramento automatico di contenuti audio, in grado di produrre risultati utilizzabili ma suscettibili di ulteriori ottimizzazioni.

Dal punto di vista numerico, i risultati possono cambiare in base al prompt utilizzato e all'audio generato dal modello. Ad esempio, nel codice è stato usato il seguente prompt:

```
prompt_text = ["An 80s pop song with
heavy synth,a groovy bassline, and
punchy drums"]
```

Nell'ultima parte del notebook viene inoltre prodotta una tabella (*table 1*)in cui sono state calcolate le seguenti metriche audio sul segnale originale e su quello processato con Demucs; in essa viene mostrato come, anche se in misura ridotta, si ha un miglioramento delle prestazioni rispetto all'audio originale .

Metrica	Originale	Migliorato
spectral_centroid	2789.3240	2773.4412
spectral_flatness	0.0444	0.0434
rms_energy	0.1310	0.1294

Table 1. Risultati Numerici

Bibliography. (Copet et al., 2023) (Lam et al., 2023) (AssemblyAI, 2025) (Meta AI, 2025) (Galope, 2024)

References

AssemblyAI. What is residual vector quantization? https://www.assemblyai.com/blog/what-is-residual-vector-quantization, 2025. Accessed: 29 August 2025.

Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., and Défossez, A. Simple and controllable music generation. *37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023. Cornell University, last revised 30 Jan 2024 (this version, v3).

Galope, S. How to use python and demucs to separate audio files. https://www.samgalope.dev/2024/07/27/how-to-use-python-and-demucs-to-separate-audio-files/, 2024.

Lam, M. W. Y., Tian, Q., Li, T., Yin, Z., Feng, S., Tu, M., Ji, Y., Xia, R., Ma, M., Song, X., Chen, J., Wang, Y., and Wang, Y. Efficient neural music generation. *arXiv Cornell University*, 2023. Speech, Audio & Music Intelligence (SAMI), ByteDance, 25 May 2023.

Meta AI. Audiocraft: Models and libraries for audio generation. https://ai.meta.com/resources/models-and-libraries/audiocraft/, 2025. Accessed: 29 August 2025.