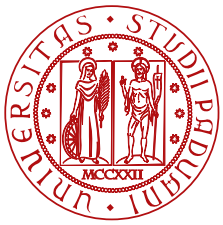

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA DELL'INFORMAZIONE

Compressione con perdite di segnali audio mediante tecniche di machine learning

Relatore
Prof. Giancarlo Calvagno

Laureanda
Giulia Tacconi

ANNO ACCADEMICO 2023-2024

Data di laurea 14/03/2024

Indice

Capitolo 1

Introduzione

Con la rivoluzione digitale la quantità di dati che bisogna gestire è notevole e destinata a crescere ulteriormente nei prossimi anni. Basti pensare ad esempio alla telefonia mobile, alle videocomunicazioni e soprattuto ad internet che, con i suoi servizi in tempo reale, richiede un massiccio e continuo scambio di dati in pochi secondi. Secondo uno studio di *Cisco* del 2021, infatti, si è visto come lo streaming online rappresenti l'82% dell'intero traffico internet. Risulta quindi evidente la necessità sempre più impellente di trasmettere, archiviare ed elaborare i dati il più efficientemente possibile.

Una delle tecniche che permette il corretto utilizzo dei servizi appena citati è la compressione dei dati. Nell'ambito delle telecomunicazioni con il termine compressione si intendono le tecniche volte a ridurre il numero di bit necessari per rappresentare in forma digitale una determinata informazione al fine di poterla trasmettere e archiviare efficientemente.

I bit necessari per rappresentare file multimediali possono essere infatti considerevoli. Ad esempio per rappresentare 2 minuti di musica in formato CD Audio (segnale campionato a 44,1 kHz con campioni di 16 bit) senza ausilio della compressione sono richiesti più di 84 milioni di bit, il che evidenzia come sarebbe impossibile scaricare musica da siti web in pochi secondi, contrariamente alle nostre abitudini ormai consolidate.

Avendo come obiettivo quello di ottimizzare la trasmissione e archiviazione di file sorge spontaneo chiedersi per quale motivo non si investa anche nel miglioramento delle capacità dei mezzi fisici a nostra disposizione. Se da un lato infatti sia la capacità di archiviazione che quella di trasmissione dei dispositivi cresce di pari passo con le innovazioni tecnologiche del settore, dall'altro però la necessità di gestire i dati si sviluppa con il doppio della velocità rispetto alla capacità di archiviazione e trasmissione dei diversi dispositivi, lasciando quindi come modo preferibile per migliorare l'efficienza lo studio di nuove tecniche per la compressione dati.

In numerosi campi applicativi della compressione di segnali vengono utilizzate sempre più le tecniche tipiche del Machine Learning, branca dell'Intelligenza Artificiale che si serve di metodi statistici per migliorare la performance di un algoritmo.

Molti degli algoritmi più innovativi, chiamati algoritmi di compressione neurale, sfruttano infatti le reti neurali per l'apprendimento end-to-end dei dati attraverso modelli generativi come i *variational autoencoders VAE* e le *Generative Adversarial Networks*.

É necessario evidenziare però che parte delle tecniche di Machine Learning utilizzate oggi sono già state impiegate a partire dal 1980, come ad esempio il K-means adottato per la quantizzazione vettoriale. La grande differenza che vi è fra questi e gli algoritmi di compressione neurale odierni risiede nella struttura e dimensione delle reti neurali che risultano essere sensibilmente migliori.

Questa tesi vuole soffermarsi sulla compressione dei segnali audio, in modo particolare su algoritmi di compressione di tipo *lossy*, ossia con perdite e andando ad analizzare infine un algoritmo innovativo sviluppato dal gruppo Meta: EnCodec.

1.1 Tipologie di compressione

Quando parliamo di tecniche di compressione stiamo in realtà facendo riferimento ad un algoritmo di compressione e ad un algoritmo di decompressione.

Considerando un generico vettore $\boldsymbol{x}$ di numeri reali, l'algoritmo di compressione prende in input $\boldsymbol{x}$ e genera la sua corrispondente rappresentazione $\boldsymbol{x}_c$ che presenta un numero di bit inferiore rispetto ai bit iniziali. L'algoritmo di decompressione invece ricostruisce il vettore $\boldsymbol{y}$ a partire da $\boldsymbol{x}_c$. In Figura 1.1 è mostrato lo schema generale di un algoritmo di compressione:

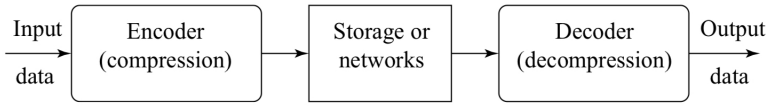

Figura 1.1: Schema generale di un algoritmo di compressione [5].

Esistono due tipi di compressione:

- senza perdite (*lossless*)
- con perdite (*lossy*)

Come suggerisce il nome, nel caso di compressione *lossless* i dati sono compressi mantenendo tutta l'informazione, quindi a partire dalla versione compressa è possibile risalire alla versione originale. In una compressione *lossy*, invece, una parte dell'informazione originale viene persa permanentemente, impedendo quindi la ricostruzione esatta del vettore originale. In generale vale:

$$\boldsymbol{y} \neq \boldsymbol{x} \tag{1.1}$$

Numerose sono le applicazioni che non possono tollerare alcuna differenza tra file originale e file ricostruito come le immagini mediche, le immagini ottenute dai satelliti, i file di testo o i dati dei registri bancari.

Al contrario nel caso di compressione di video e segnali audio destinati allo streaming online è accettabile una ricostruzione non esatta del segnale di input. Ciò è dovuto al fatto che questo tipo di file multimediali sono destinati all'essere umano, il quale possiede abilità percettive limitate che vengono quindi sfruttate per ottenere un tasso di compressione maggiore mantenendo comunque una buona qualità e un certo grado di fedeltà con il file originale.

1.2 Performance di un algoritmo

Un algoritmo di compressione è spesso valutato in base alla percentuale di bit che vengono compressi. Si parla quindi spesso di diminuzione del bitrate, numero di cifre binarie impiegate per immagazzinare un secondo di informazione, e di tasso di compressione.

Dato il numero di bit totali per rappresentare un file prima della compressione $\mathcal{B}_0$ e il numero totale di bit per rappresentare il file dopo la compressione $\mathcal{B}_1$, si definisce tasso di compressione il loro rapporto:

$$\textit{tasso di compressione} = \frac{\mathcal{B}_0}{\mathcal{B}_1} \tag{1.2}$$

In generale il tasso di compressione per i codec[1] che effettuano una compressione audio con perdite è di $\simeq 10$, valore per il quale il segnale compresso è praticamente indistingubile dall'originale all'orecchio umano. In Figura 1.2 è possibile vedere un confronto di dimensioni fra segnale compresso e la sua versione originale.

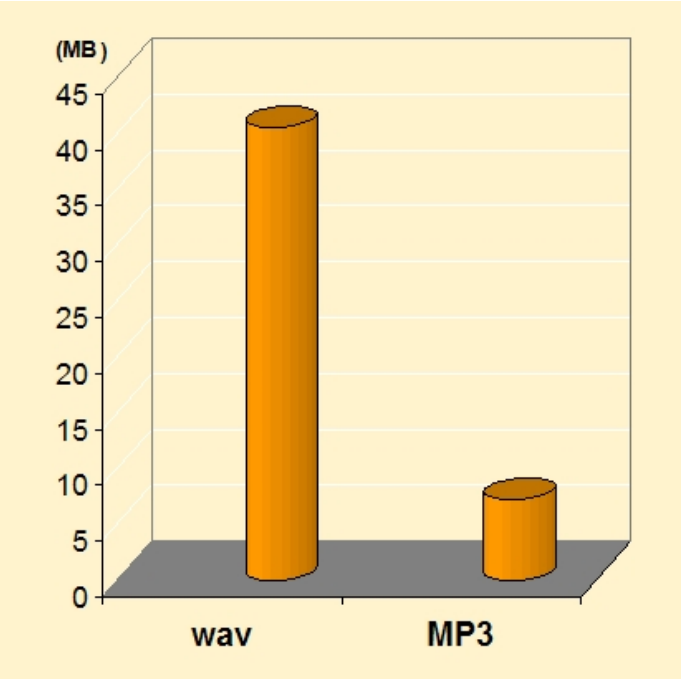

Figura 1.2: Dimesioni file formato WAV (segnale non compresso) a confronto con segnale compresso in formato MP3 [11].

Per quanto riguarda la compressione *lossy* un altro parametro fondamentale per analizzare l'efficacia di un algoritmo è la distorsione, ossia quanto il file ricostruito differisca da quello originale. Per misurare la distorsione tipicamente si utilizza il *Mean Squared Er-*

[1] Software che si occupa della codifica e decodifica di un segnale.

ror (MSE)[2] che, data una sequenza di variabili aleatorie X_n e la corrispondente sequenza ricostruita Y_n, per una sequenza di lunghezza N è definito come:

$$D = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}[(X_n - Y_n)^2] \tag{1.3}$$

Spesso si è inoltre interessati a misurare il rapporto fra la potenza del segnale utile e la potenza del segnale di rumore, detto *Signal to Noise Ratio (SNR)*[3], spesso misurata in *decibel (dB)*:

$$SNR_{dB} = 10 \log_{10} \frac{P_{segnale}}{P_{rumore}} \tag{1.4}$$

L'*SNR* misura quindi la qualità del segnale compresso, in quanto la sua discrepanza con il segnale originale è spesso definita come distorsione.

I parametri appena introdotti costituiscono un utile strumento matematico per valutare la fedeltà di un segnale compresso rispetto al segnale originale, tuttavia non riescono a stabilire come il segnale ricostruito venga percepito, nel caso ad esempio di segnali audio, dall'orecchio umano. Per questo motivo è necessario trovare un modello matematico che possa descrivere la percezione umana al fine di poter prendere in considerazione anche questo importante aspetto sia per l'analisi delle performance che per la fase di progettazione di un algoritmo. Ogni codec infatti punta a cercare il giusto trade-off fra diminuzione del bitrate e introduzione di distorsione tale per cui la qualità del segnale percepita non risulti eccessivamente compromessa.

[2] Errore Quadratico Medio.
[3] Rapporto segnale/rumore.

Capitolo 2

Principi della compressione lossy di segnali audio

Il Moving Picture Expert Group-1/2 Audio Layer 3, conosciuto più comunemente con il nome di MP3 è l'algoritmo di compressione audio più conosciuto. Sviluppato e rilasciato nel 1992 dall'istituto di ricerca tedesco Fraunhofer IIS, MP3 è presto diventato il formato predefinito per la distribuzione di musica online grazie alla sua capacità di offrire agli utenti una qualità di riproduzione del file audio elevata, occupando molta meno memoria rimanendo al contempo fedele al file originale non compresso. Si può quindi dire che MP3 sia stato il primo algoritmo ad aver gettato le basi per tutte le tecniche e formati di compressione audio che oggi conosciamo, come ad esempio l'Advanced Audio Coding (AAC) e Ogg Vorbis.

2.1 Progettazione di un algoritmo di compressione

Ogni algoritmo di compressione dati può essere visto come la composizione di due fasi:

- la *modellizzazione* è la prima fase in cui si ricavano informazioni su eventuali ridondanze presenti nel segnale e si descrivono mediante un modello matematico;
- la *codifica* è invece la seconda fase nella quale le informazioni identificate precedentemente sono convertite in un formato più efficiente, generalmente utilizzando una sequenza binaria.

MP3 è un algoritmo di compressione audio di tipo *lossy* e come la maggior parte di questi sfrutta il modello psicoacustico per attenuare o addirittura eliminare dal segnale tutte le componenti che sono inudibili all'orecchio umano. Successivamente sfrutta la quantizzazione per ridurre le dimensioni del segnale datogli in input.

2.2 Psicoacustica

La psicoacustica è la disciplina che studia come il suono viene percepito dall'essere umano sia a livello fisiologico, basato sulla struttura dell'orecchio, che a livello psicologico, ossia di come il suono viene elaborato nel nostro cervello. Il modello psicoacustico adoperato nella compressione di segnali audio con perdite si basa sugli studi svolti nel capire come sfruttare le limitazioni acustiche a livello percettivo dell'uomo per selezionare le informazioni che devono essere codificate e quelle che invece possono essere scartate.

L'orecchio umano può udire i suoni nell'intervallo dai 20 Hz ai 20 kHz se prendiamo in considerazione la frequenza, mentre il range di udibilità a livello di intensità del suono va dai 0 dB ai 120 dB.

2.2.1 Relazioni di pari intensità sonora

Diversi studi hanno mostrato come l'intensità dipenda dalla frequenza oltre che dal livello effettivo del suono, infatti, ad esempio, un segnale acustico a 1 kHz con intensità di 20 dB, viene percepito allo stesso modo di un segnale a 50 Hz con livello d'intensità pari a 50 dB. Il legame fra intensità e frequenza costituisce un importante aspetto per la psicoacustica ed è solitamente rappresentato come un insieme di curve che vengono percepite allo stesso livello di volume, chiamate *curve di Fletcher-Munson* (1933), perfezionate successivamente da Robinson e Dadson nel 1956. (Vedi Figura 2.1).

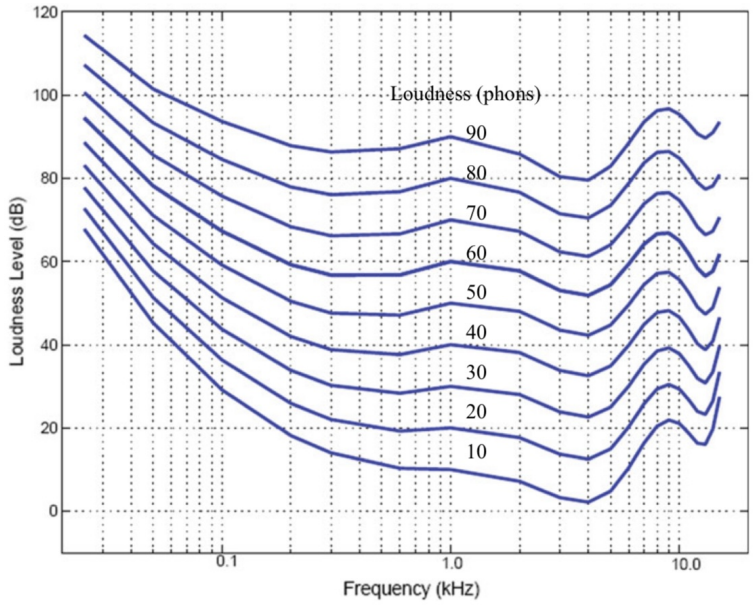

Figura 2.1: Curve di pari intensità sonora di Fletcher-Munson rimisurate da Robinson e Dadson [5].

La curva del grafico di maggiore interesse è quella che rappresenta la soglia uditiva, ossia la curva del livello di pressione sonora (SPL) che delimita il confine tra suoni udibili e inudibili a diverse frequenze, come si può vedere in Figura 2.2.

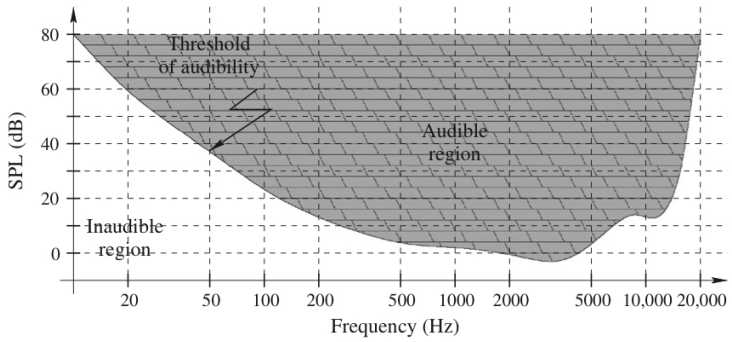

Figura 2.2: Curva della soglia uditiva [10].

2.2.2 Effetti di mascheramento

Un altro interessante fenomeno audio è il mascheramento nel quale un suono riesce a nascondere o addirittura eliminare la percezione di un altro. Esistono due tipi di mascheramento:

- lo *spectral masking* (mascheramento di frequenza);
- il *temporal masking* (mascheramento temporale).

Spectral Masking

Il mascheramento di frequenza, chiamato anche mascheramento simultaneo poichè si verifica quando si hanno due suoni contemporaneamente, avviene quando un suono puro viene nascosto da un suono detto rumore quando quest'ultimo si trova in un piccolo intervallo di frequenze attorno al tono puro, chiamato banda critica. Si è quindi interessati a capire quanto rumore è tollerato prima che non sia più possibile distinguere il rumore dal tono preso in considerazione.

Si può vedere come la soglia di udibilità non sia assolutamente fissa, bensì questa tende ad alzarsi quando vi sono più suoni contemporaneamente, in altre parole un tono puro ad una certa frequenza aumenta la soglia nella banda critica attorno a quella frequenza.

In Figura 2.3 possiamo vedere come un tono puro a 1 kHz abbia aumentato la soglia di audibilità rendendo impercettibile il tono a 1500 Hz. Questo fenomeno è molto rilevante per gli algoritmi di compressione *lossy*. Come vedremo meglio nella sezione successiva

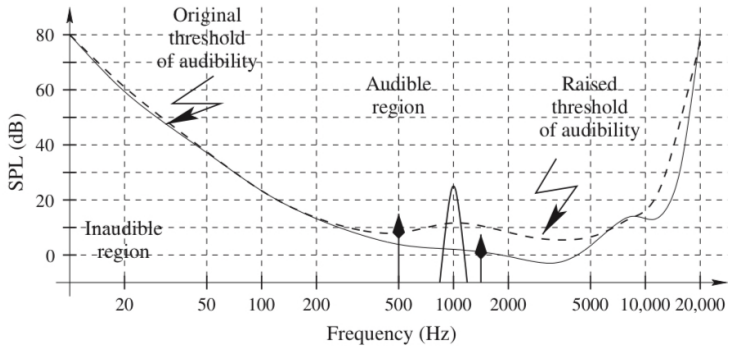

Figura 2.3: Variazione della curva della soglia uditiva [10].

di questo capitolo, infatti, nella codifica con perdite si utilizza la quantizzazione, un processo nel quale il segnale in uscita è ottenuto da un segnale d'ingresso al quale si somma il rumore di quantizzazione additivo. L'obiettivo è quindi scegliere un passo di quantizzazione tale che il rumore di quantizzazione rimanga sempre sotto la soglia di udibilità in modo tale da renderlo impercettibile. Il fenomeno della variazione della curva della soglia uditiva viene sfruttato soprattutto per introdurre più rumore di quantizzazione laddove la curva aumenta di livello.

Temporal Masking

L'effetto di mascheramento temporale avviene quando un suono aumenta la soglia di udibilità per un breve intervallo che precede e segue il suono stesso.

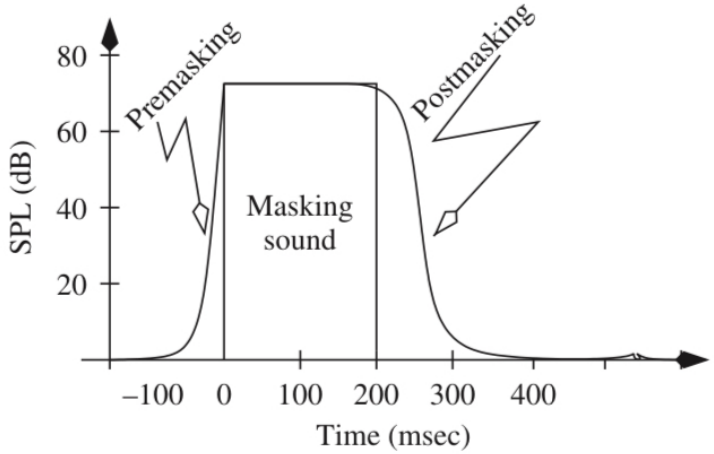

Figura 2.4: Variazione della soglia uditiva nel dominio del tempo [10].

Il fatto che i suoni che si verificano prima del tono mascherante possano essere nascosti

può essere visto come prova della natura interferenziale della percezione uditiva umana. Come si può notare in Figura 2.4, se il tono mascherato precede il suono mascherante si parla di *premasking*, al contrario se il tono mascherato è succesivo si verifica il cosiddetto *postmasking*. É da notare come il *postmasking* abbia durata ben più lunga del *premasking*.

2.2.3 Modello psicoacustico

Il modello psicoacustico non è univoco, tuttavia le diverse versioni utilizzate negli algoritmi di compressione con perdite differiscono solo per qualche dettaglio ed è quindi possibile presentare un modello generale. Il primo passaggio del modello psicoacustico è quello di ottenere il profilo spettrale del segnale che si vuole codificare. Il segnale audio in ingresso viene scomposto in finestre temporali ed elaborato da un banco di filtri digitali al fine di rappresentarlo in molteplici sottobande nello spettro audio. Per ogni banda spettrale viene poi calcolato il livello di pressione sonora (SPL) e se l'algoritmo utilizza un approccio a sottobande l'SPL della banda è ricavato dall'SPL di ciascun coefficiente di frequenza X_k. Il secondo passo è quello di distinguere le componenti tonali in quanto presentano effetti diversi di mascheramento rispetto alle altre componenti. La presenza di componenti tonali è determinata andando in primo luogo ad osservare i massimi locali, dove un massimo locale si trova in k se $|X_k|^2 > |X_{k-1}|^2$ e $|X_k|^2 > |X_{k+1}|^2$.

Un massimo locale è una componente tonale se vale:

$$20log_{10}\frac{|X_k|}{|X_{k+j}|} \geq 7 \tag{2.1}$$

dove il valore j dipende dalla frequenza. Una volta identificati i mascheranti tonali questi vengono rimossi da ogni banda critica e la potenza delle rimanenti linee spettrali viene sommata per ottenere il livello di mascheramento non tonale. Una volta che tutti i toni mascheranti sono stati correttamente individuati vengono rimossi tutti quelli che presentano un valore di SPL sotto la soglia di udibilità. Inoltre, tra tutti i toni mascheranti che sono estremamente vicini tra loro a livello di frequenza, viene eliminato il tono con ampiezza minore. L'effetto dei rimanenti suoni mascheranti è ottenuto tramite una funzione di diffusione che modella il mascheramento in frequenza. Infine è possibile creare le soglie di mascheramento che vengono utilizzate nel processo di codifica.

2.3 Quantizzazione

L'elemento cardine della codifica di tipo *lossy* è la quantizzazione che è un processo che permette di rappresentare un insieme di valori molto grande, potenzialmente infinito (segnali analogici), con un insieme di valori di dimensioni nettamente inferiori. Un semplice esempio per capire il funzionamento della quantizzazione è il seguente: data una sorgente che genera numeri reali nell'intervallo $[-5{,}5]$ un possibile schema di quantizzazione è costituito dal rappresentare ogni valore generato dalla sorgente con il corrispettivo numero intero più vicino. Ad esempio 2.37 e 2.43 verrebbero arrotondati a 2 e 3.14 verrebbe rappresentato dal valore 3. Per quanto banale, l'esempio appena mostrato evidenzia come il processo di quantizzazione sia irreversibile, in quanto effettuando l'operazione di arrotondamento il valore originale generato dalla sorgente è perso definitivamente.

I valori d'ingresso e d'uscita di un quantizzatore possono essere sia scalari che vettoriali, per questo motivo viene spesso fatta la distinzione fra *quantizzazione scalare* e *quantizzazione vettoriale*. Inoltre la quantizzazione si può suddividere in due tipologie in base alla distribuzione dei livelli di quantizzazione lungo l'intervallo del segnale che si vuole quantizzare che può essere *uniforme* o *non-uniforme*.

2.3.1 Quantizzazione scalare

Il processo di quantizzazione può essere descritto dalla funzione:

$$\mathcal{Q} : \mathbb{R} \rightarrow \mathcal{A}_q \tag{2.2}$$

dove ad ogni valore d'ingresso che appartiene all'insieme dei numeri reali $\mathbb{R}$ è associato il corrrispettivo valore dell'alfabeto di quantizzazione $\mathcal{A}_q$ che viene a sua volta rappresentato tramite una *codeword* **c** di b bit.
In altre parole la quantizzazione determina una partizione dell'asse reale in regioni R_i, con $i = 0, ..., L-1$, tali che $\mathbb{R} = \bigcup_{i=0}^{L-1} R_i$ e $R_i \cap R_j \neq \emptyset$ per $i \neq j$.
Le regioni di quantizzazione sono tipicamente definite nel seguente modo:

$$R_i = (v_i, v_{i+1}] \quad \text{con} \quad i = 0, ..., L-1 \tag{2.3}$$

dove $v_0 = -\infty$ e $v_L = +\infty$. Le restanti *soglie di quantizzazione* v_i non sono fisse e devono invece essere determinate.
Una rappresentazione grafica di un generico quantizzatore è illustrata in Figura 2.5.

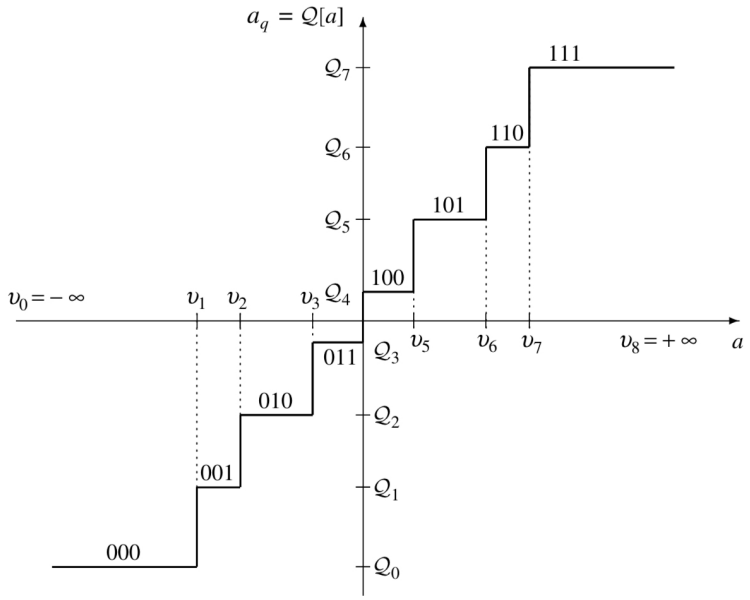

Figura 2.5: Esempio di quantizzazione a $L = 8$ livelli [8].

Si può notare nell'esempio in figura come ad ogni valore dell'alfabeto $\mathcal{A}_q$ venga assegnata una sequenza binaria di $\lceil \log_2 L \rceil$ bit.

Il processo di campionamento e quantizzazione di un segnale analogico, come nel caso di un segnale audio, è chiamato *Pulse Code Modulation (PCM)*. La Figura 2.6 mostra la modulazione ad impulsi codificati di un segnale.

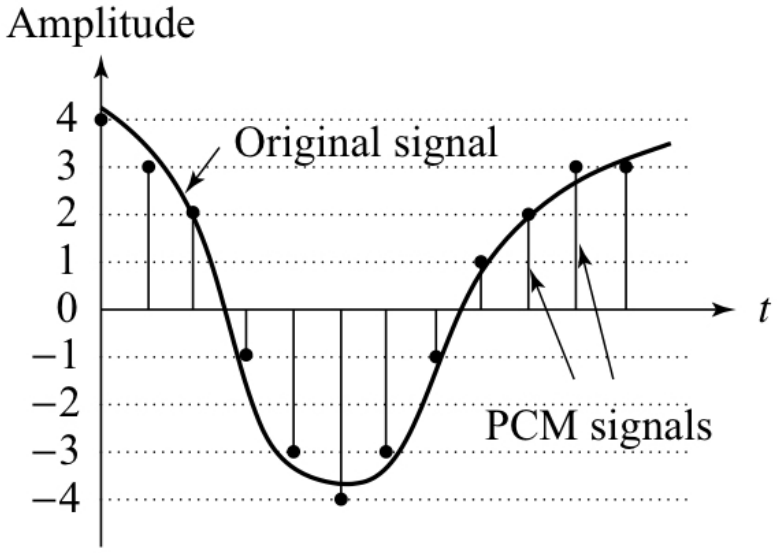

Figura 2.6: PCM [5].

2.3.2 Quantizzazione scalare uniforme

Un quantizzatore scalare uniforme partiziona il dominio dei valori del segnale analogico in intervalli della stessa grandezza. In altre parole valgono le seguenti relazioni:

$$\begin{aligned} v_i - v_{i-1} &= \Delta, \quad i = 2, ..., L-1 \\ \mathcal{Q}_i - \mathcal{Q}_{i-1} &= \Delta, \quad i = 1, ..., L-1 \end{aligned} \tag{2.4}$$

dove Δ è il *passo di quantizzazione*.
I quantizzatori scalari uniformi si suddividono in due tipi: *midrise* e *midtread*. I valori quantizzati possono essere calcolati nel seguente modo:

$$\begin{aligned} \mathcal{Q}_{midrise}(x) &= \lceil x \rceil + 0.5\Delta \\ \mathcal{Q}_{midtread}(x) &= \lfloor x + 0.5\Delta \rfloor \end{aligned} \tag{2.5}$$

La quantizzazione *midrise* (Fig. 2.7) utilizza un numero di livelli pari, mentre quella *midtread* (Fig. 2.8) un numero dispari. La differenza più importante però è quella che solo il *midtread* include il valore 0 fra i suoi livelli di output e per questo motivo viene spesso chiamato *dead-zone quantizer*. A causa di questa caratteristica i quantizzatori *midtread* sono impiegati nella codifica di segnali audio nella quale il valore 0 rappresenta i periodi di silienzio.

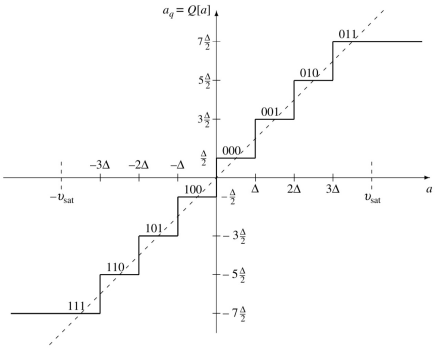

Figura 2.7: Quantizzatore *midrise* [8].

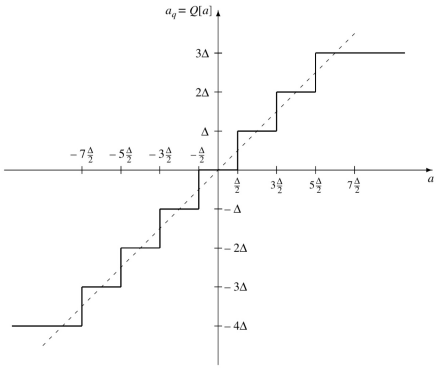

Figura 2.8: Quantizzatore *midtread* [8].

L'obiettivo nella progettazione di un quantizzatore uniforme è quello di trovare il *passo di quantizzazione* Δ che minimizzi la distorsione del segnale di input e il numero di livelli di quantizzazione L. Andando ad esaminare la performance di un quantizzatore a L livelli, considerando di avere in ingresso un segnale uniformemente distribuito in

$[-v_{sat}, v_{sat}]$, il bitrate del quantizzatore, ossia il numero di bit necessari per codificare L livelli di uscita è:

$$R = \lceil log_2 L \rceil \tag{2.6}$$

Il passo di quantizzazione Δ è definito come:

$$\Delta = \frac{2v_{sat}}{L} \tag{2.7}$$

in quanto i valori d'ingresso appartengono all'intervallo $[-v_{sat}, v_{sat}]$. Per un segnale limitato, l'errore generato dalla quantizzazione è detto *distorsione granulare*, se il segnale è invece illimitato viene chiamata *distorsione di overload*.

$$D_{granulare} = \sum_{i=0}^{L-1} \int_{v_i}^{v_{i+1}} (x - \mathcal{Q}_i)^2 \frac{1}{2v_{sat}} dx \tag{2.8}$$

Poichè i valori di ricostruzione $\mathcal{Q}_i$ sono i punti medi di ogni intervallo, l'errore di quantizzazione deve appartenere a $[-\frac{\Delta}{2}, \frac{\Delta}{2}]$, come mostrato in Figura 2.9.

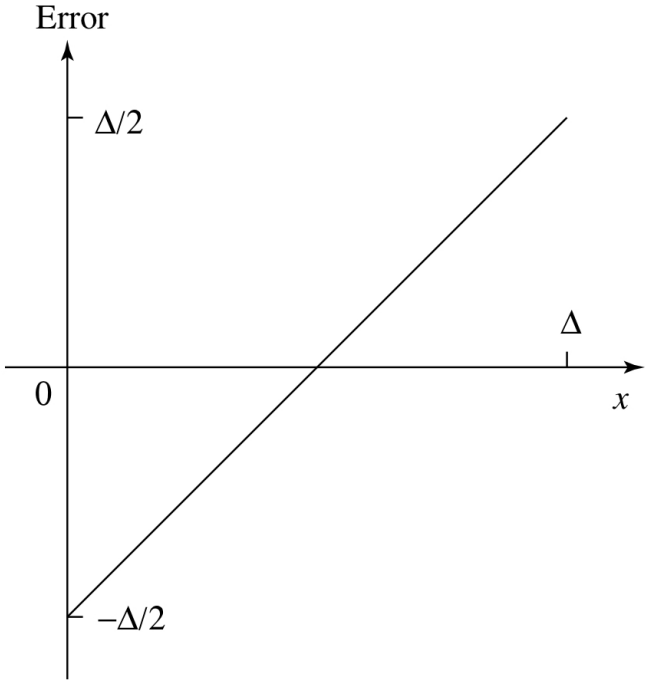

Figura 2.9: Errore di quantizzazione per un segnale uniformemente distribuito [5].

L'MSE, avendo un errore di quantizzazione uniformemente distribuito, equivale alla varianza σ_d^2 dell'errore di quantizzazione calcolato nell'intervallo $[0, \Delta]$. Definendo

l'errore in funzione di x come $e(x) = x - \Delta/2$ si ha:

$$\begin{aligned}\sigma_d^2 &= \frac{1}{\Delta}\int_0^{\Delta}(e(x)-x)^2dx \\ &= \frac{1}{\Delta}\int_0^{\Delta}(x-\Delta/2-x)^2dx \\ &= \frac{\Delta^2}{12}\end{aligned} \tag{2.9}$$

Analogamente, la varianza del segnale aleatorio x è $\sigma_x^2 = (2v_{sat})^2/12$, il che porta ad un SQNR[1]:

$$\begin{aligned}SQNR &= 10\log_{10}(\frac{\sigma_x^2}{\sigma_d^2}) \\ &= 10\log_{10}L^2 \\ &= 20b\log_{10}2 \\ &= 6.02b \quad [dB]\end{aligned} \tag{2.10}$$

dove si è utilizzata la relazione $L = 2^b$. Da questa formula otteniamo l'importante risultato che l'aumento di un bit porta ad un incremento del rapporto segnale-rumore di $6.02\,dB$

2.3.3 Quantizzazione non-uniforme

Se il segnale da quantizzare non è uniformemente distribuito, un quantizzatore uniforme risulta essere inefficiente. Aumentare il numero di livelli di quantizzazione nella regione dove il segnale è densamente distribuito può diminuire notevolmente la distorsione granulare. Inoltre, senza dover aumentare il numero totale di livelli di quantizzazione, è possibile allargare la regione in cui il valore del segnale è poco probabile. I *quantizzatori non-uniformi* quindi non presentano passo di quantizzazione Δ costante. Vi sono due approcci per costruire *quantizzatori non-uniformi*: l'algoritmo di Lloyd-Max e la tecnica di compressione-espansione.

Algoritmo di Lloyd-Max

L'algoritmo di Lloyd-Max è un processo iterativo che permette di ottenere un quantizzatore ottimo. Data la funzione di densità di probabilità (PDF)[2] $p_X(x)$ del segnale in ingresso,

[1] Signal-to-quantization-noise ratio.
[2] Probability Density Function.

l'errore di quantizzazione è definito nel seguente modo:

$$D_{eq} = \sum_{i=0}^{L-1} \int_{v_i}^{v_{i+1}} (\mathcal{Q}_i - x)^2 p_X(x) dx \tag{2.11}$$

L'obiettivo dell'algoritmo è quello di minimizzare la distorsione totale ponendo la derivata dell'eq. (2.11) uguale a 0. Differenziando rispetto a $\mathcal{Q}_i$ si ottengono i rispettivi valori quantizzati:

$$\mathcal{Q}_i = \frac{\int_{v_i}^{v_{i+1}} x\, p_X(x) dx}{\int_{v_i}^{v_{i+1}} p_X(x) dx} \tag{2.12}$$

Si può quindi facilmente notare come i livelli di quantizzazione ottimi siano il baricentro dell'intervallo $[v_i, v_{i+1}]$. Differenziando invece rispetto alle soglie v_i e ponendo sempre il risultato uguale a 0 si ricava:

$$v_i = \frac{\mathcal{Q}_{i-1} + \mathcal{Q}_i}{2} \tag{2.13}$$

Quindi le soglie di quantizzazione ottime sono il punto medio tra due livelli adiacenti. Risolvendo queste due equazioni simultaneamente e procedendo iterazione dopo iterazione si ottiene il quantizzatore ottimo di Lloyd-Max.

Tecnica di compressione-espansione

Come suggerisce il nome, il segnale d'ingresso è mappato da una *funzione di compressione* G che modifica l'ampiezza del segnale al fine di rappresentare con maggior precisione i segmenti del segnale più rilevanti. Il segnale viene successivamente quantizzato mediante un quantizzatore uniforme e dopo la trasmissione una *funzione d'espansione* G^{-1} ne ripristina la sua dinamica originale (vedi Fig. 2.10).

Questo metodo di *quantizzazione non-uniforme* è ampliamente impiegato nella compressione del segnale vocale[3] finalizzata alla trasmissione lungo le linee telefoniche. In questo particolare caso infatti con le due funzioni G e G^{-1} si cerca di ridurre l'effetto della quantizzazione sui segmenti a bassa ampiezza del segnale, punti nei quali la quantizzazione è più evidente all'udito umano, in quanto maggiormente sensibile alle variazioni.

[3] Speech Coding: sottocategoria della compressione audio che si focalizza sui segnali contenenti il parlato.

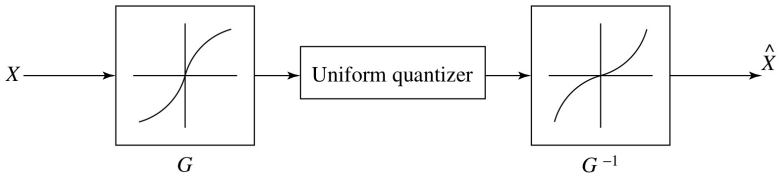

Figura 2.10: Quantizzazione mediante compressione-espansione [5]

.

2.3.4 Quantizzazione vettoriale

Il vantaggio della quantizzazione vettoriale rispetto alla quantizzazione scalare risiede soprattutto nella maggior efficienza[4]. La quantizzazione vettoriale infatti è in grado di individuare e sfruttare le correlazioni che spesso, come nei segnali audio, esistono fra i campioni adiacenti che vengono quindi concatenati assieme per formare i vettori che devono essere quantizzati. Prendiamo ad esempio in considerazione L campioni consecutivi che uniti costituiscono il vettore di input del quantizzatore L-dimensionale. Sia l'*encoder*[5] che il *decoder*[6] del quantizzatore vettoriale presentano un insieme di vettori L-dimensionali, i *code-vectors*, che formano il *codebook* del quantizzatore. I *code-vectors* hanno la funzione di rappresentare i vettori di input e ad ognuno di questi è assegnato un indice, tipicamente binario, di lunghezza inferiore rispetto a L. Nell'*encoder* i vettori di input vengono confrontati con ogni *code-vector* al fine di trovare quello più vicino al vettore in entrata. Una volta scelto il vettore del *codebook* migliore viene trasmesso il suo indice al *decoder* che tramite un semplice confronto riesce a determinare il vettore per la ricostruzione del segnale. Il punto chiave della compressione risiede dunque nella trasmissione dell'indice del *code-vector* selezionato che essendo di dimensioni inferiori presenta meno informazione rispetto ai valori effettivi dei campioni. Il processo di quantizzazione vettoriale è mostrato in Figura 2.11.

[4]Con maggior efficienza si intende minor distorsione (misurata tramite MSE e SNR) a un bit rate fissato.
[5]Parte del quantizzatore che si occupa di trasformare il segnale analogico in forma digitale.
[6]Parte del quantizzatore che converte i valori quantizzati e ricostruisce il segnale analogico originale.

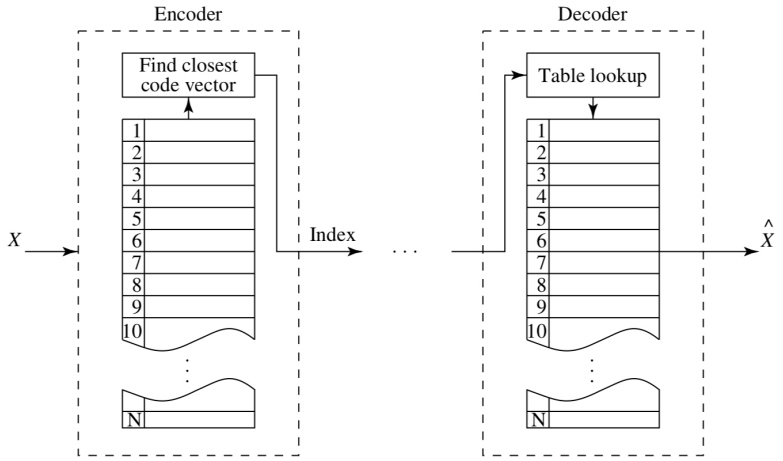

Figura 2.11: Quantizzazione vettoriale [5].

Algoritmo di Linde, Buzo, Gray

Nella fase di progettazione di un quantizzatore vettoriale l'aspetto a cui bisogna prestare maggiormente attenzione è sicuramente la determinazione del *codebook* $\mathcal{C}$. Tra i diversi metodi utilizzabili per generare $\mathcal{C}$ quello impiegato principalmente, conosciuto come algoritmo di Linde, Buzo, Gray (LBG), permette di ottenere il *codebook* ottimale per la quantizzazione.

Dato il *training set* $\mathcal{X} = \{\boldsymbol{x_1}, ..., \boldsymbol{x_N}\}$ dove ogni vettore $\boldsymbol{x_i}$ ha dimensione D, costituito dai vettori di input del quantizzatore, l'algoritmo iterativo LBG si compone dei seguenti passi:

1. si inizializza il *codebook* $\mathcal{C}_0 = \{\boldsymbol{c_1}, ..., \boldsymbol{c_m}\}$ dove ogni *code-vector* c_j ha anch'esso dimensione D. Definisco inoltre una soglia ϵ e pongo l'indice delle iterazioni $k = 0$.

2. vengono determinate le regioni di decisione:

$$\mathcal{R}_j^k = \{\boldsymbol{x}_i : d(\boldsymbol{x}_i, \boldsymbol{c}_j) < d(\boldsymbol{x}_i, \boldsymbol{c}_n) \forall\, n \neq j\} \quad j = 1, ..., M \tag{2.14}$$

   e si calcola la distorsione sul *training set* $\mathcal{X}$:

$$D^k = \frac{1}{N} \sum_{j=1}^{M} \sum_{\boldsymbol{x_i} \in \mathcal{R}_j^k} \|\boldsymbol{x_i} - \boldsymbol{c_j}\|^2 \tag{2.15}$$

3. se

$$\frac{D^k - D^{k-1}}{D^k} < \epsilon \tag{2.16}$$

   ho trovato il *codebook* ottimo e l'algoritmo si ferma, altrimenti continua.

4. $k = k + 1$.

$$\boldsymbol{c}_j^k = \frac{1}{|\mathcal{R}_j^{k-1}|} \sum_{\boldsymbol{x}_i \in \mathcal{R}_j^{k-1}} \boldsymbol{x}_i \quad k = 1, ..., M \tag{2.17}$$

trova i valori $\boldsymbol{c_j}^k$ per $j = 1, ..., M$ che siano il valor medio degli elementi di ogni regione di quantizzazione $\mathcal{R}_j^{k-1}$ e ritorna al secondo step.

L'algoritmo LBG può essere visto come un'estensione dell'algoritmo K-means tipico del machine learning e sviluppato per il riconosimento di pattern.

Capitolo 3

EnCodec

EnCodec è un algoritmo sviluppato nel 2022 dall'impresa statunitense Meta. EnCodec è un algoritmo di compressione con perdite di segnali audio che sfrutta l'intelligenza artificiale per comprimere file audio al fine di poterli condividere facilmente e con maggior velocità in rete. Se da un lato alcuni studi sono stati precedentemenenti condotti per la compressione di segnali vocali, questo algoritmo è il primo a sfruttare le tecniche del Deep Learning per la compressione di segnali audio stereo campionati a $48kHz$ che rappresenta lo standard per la distribuzione musicale.

3.1 Modello matematico

Un segnale audio a durata d può essere rappresentato da una sequenza $\boldsymbol{x} \in [-1,1]^{C_a \times T}$ dove C_a è il numero di canali audio e $T = d \cdot f_{sr}$ è il numero di campioni audio alla frequenza di campionamento f_{sr}. Il modello dell'algoritmo EnCodec può essere visto come la composizione di tre elementi principali sequenzali:

- La *rete del codificatore* E che presenta come input un segnale audio e restituisce in output la sua rappresentazione latente in forma vettoriale $\boldsymbol{z}$.
- Il *livello di quantizzazione* Q che produce la rappresentazione compressa $\boldsymbol{z_q}$ tramite una versione modificata dell'algoritmo di quantizzazione vettoriale.
- La *rete del decodificatore* G che ricostruisce il segnale rappresentato nel dominio del tempo, $\hat{\boldsymbol{x}}$, a partire da $\boldsymbol{z_q}$.
  $\hat{\boldsymbol{x}}$ viene successivamente confrontato con il segnale audio originale utilizzando un *discriminatore* che calcola la differenza fra i due, chiamata *discriminator-generator losses*, ossia perdite tra discriminatore e generatore del segnale.

Il modello viene allenato con approccio end-to-end[1] per minimizzare le perdite che avvengono nel processo di compressione al fine di ottenere un segnale audio in output che sia il più simile possibile all'originale. Oltre alle perdite dovute al discriminatore infatti, il modello tiene conto anche delle differenze riscontrate andando a confrontare le forme d'onda[2] e lo spettrogramma di Mel[3] del segnale ricostruito con quello originale. Uno schema grafico del modello presentato è presentato in Figura 3.1.

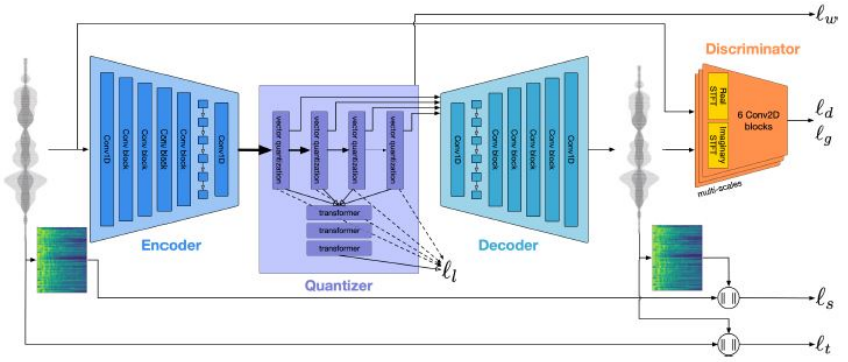

Figura 3.1: Architettura di EnCodec [4].

Il modello è allenato con le perdite di ricostruzione l_s e l_t e con le adversarial losses (l_g per il generatore e l_d per il discriminatore). La perdita dovuta alla quantizzazione vettoriale l_w si applica solo all'encoder.

3.1.1 Architettura dell'encoder e decoder

EnCodec è un codec la cui architettura si basa su reti neurali convoluzionali (CNN) di streaming, ovvero analizza la rappresentazione latente del segnale d'ingresso in tempo reale e in modo sequenziale, considerando una finestra temporale per volta.

L'encoder E è costituito da una serie di passaggi progressivi. La sequenza $\boldsymbol{x} \in [-1,1]^{C_a \times T}$ viene inizialmente analizzata da un blocco convoluzionale monodimensionale che rileva le caratteristiche principali del segnale che bisogna comprimere, come ad esempio variazioni nel volume e cambi di tonalità. I successivi blocchi convoluzionali sono costituiti a loro volta da due parti: un'unità residuale seguita da un livello di down-sampling. L'unità residuale è una componente che facilita il flusso d'informazione del segnale da codificare, mentre lo strato di down-sampling sfrutta la *strided convolution*[4]

[1] Per *end-to-end* si intende un processo del deep learning nel quale tutte le parti del modello sono allenate simultaneamente e non sequenzialmente.

[2] Rappresentazione grafica del suono nel dominio del tempo.

[3] Rappresentazione grafica del suono nel dominio delle frequenze.

[4] Variante della convoluzione che aumenta il passo con cui viene spostata la finestra di convoluzione, andando così a diminuire la dimensione della matrice in ingresso.

per ridurre gradualmente le dimensioni di $\boldsymbol{x}$, andando a selezionare ulteriori informazioni essenziali del segnale originale. I blocchi convoluzionali sono poi seguiti da una LSTM[5] che permette di non distorcere la struttura temporale del segnale. Infine l'ultimo blocco convoluzionale restituisce la rappresentazione vettoriale $\boldsymbol{z} \in \mathbb{R}^{DxT}$ di $\boldsymbol{x}$, dove D è il numero di canali d'uscita, inferiore rispetto al numero di canali del segnale originale C_a.

Il decoder G presenta la stessa struttura dell'encoder, con l'unica differenza che utilizza una diversa tecnica di convoluzione, la *transposed convolution*[6], per ricostruire la versione compressa del segnale audio.

3.1.2 Quantizzazione vettoriale residuale

Come visto nella Sezione 2.3.4, un quantizzatore vettoriale presenta un *codebook* contenente un certo numero di *code-vectors* della stessa lunghezza dei vettori che sono forniti in ingresso. La fase di compressione consiste nell'andare a confrontare ogni vettore in ingresso con quelli del *codebook* ed estrarre l'indice del *code-vector* che maggiormente assomiglia al vettore preso in considerazione per diminuirne le dimensioni. Il limite dell'avere un singolo quantizzatore risiede nel fatto che sarebbe necessario un *codebook* esponenzialmente grande per poter rappresentare accuratamente il segnale compresso.

La *quantizzazione vettoriale residuale (RVQ)* offre un'efficace soluzione a questo problema. Essa infatti è in grado di ottenere una progressiva approssimazione sempre più accurata del vettore d'ingresso grazie ad una serie di quantizzatori, ciascuno dotato del proprio *codebook*, che scompongono il processo di quantizzazione su più livelli.

L'idea di base è molto semplice (vedi Fig. 3.2): il *codebook* del primo strato permette di effettuare la cosiddetta *quantizzazone del prim'ordine*. Successivamente si calcolano i residui, ossia la differenza fra il vettore che rappresenta il segnale originale e il vettore quantizzato, che vengono ulteriormente quantizzati utilizzando un secondo *codebook* e così per tutti gli altri livelli.

[5] Long Short-Term Memory.

[6] Gli strati convoluzionali generano una mappa delle caratteristiche del vettore in uscita più grande rispetto a quella che riceve in ingresso.

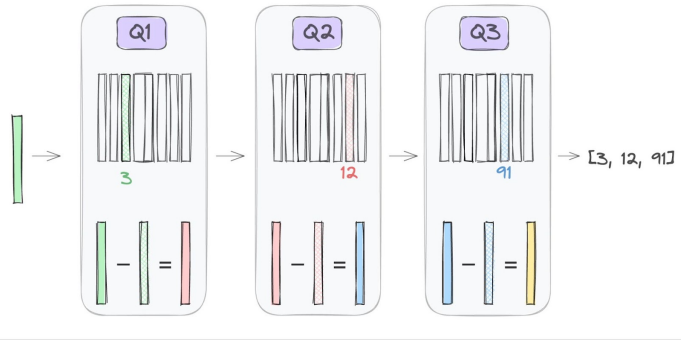

Figura 3.2: RVQ [9].

Nel nostro caso il risultato finale della quantizzazione è una matrice $\boldsymbol{z_q} \in \mathbb{R}^{N_q x T}$ dove N_q è il numero di *codebook* presenti nei diversi blocchi del quantizzatore.

3.1.3 Transformer model e codifica entropica

All'interno del quantizzatore Q è presente anche un transformer model precedentemente addestrato per migliorare la velocità del processo di quantizzazione. Il transformer è infatti capace di estrarre rapidamente informazioni su eventuali dipendenze esistenti all'interno del segnale audio, il quale viene dato in ingresso al transformer tramite la sua rappresentazione nel dominio delle frequenze MS STFT[7]. L'uscita del transformer è un insieme di logit, ossia dei valori che indicano la probabilità che un vettore si assomigli maggiormente ad un determinato *code-vector* piuttosto che ad un altro. In questo modo il transformer aiuta il quantizzatore nella scelta del giusto *code-vector* da utilizzare per la compressione.

Dopo la fase di quantizzazione si può utilizzare la codifica entropica per migliorare ulteriormente l'efficienza della compressione. In questo caso si sfruttano i logit calcolati dal transformer per assegnare codici binari più corti ai *code-vector* più frequenti e sequenza binarie più lunghe per i *code-vector* meno probabili.

3.1.4 Obiettivo d'addestramento

La fase d'addestramento è fondamentale per tutti i modelli che si avvalgono di reti neurali, in quanto è in questo contesto che il modello impara ad eseguire un compito specifico, in questo caso la compressione di segnali audio. L'obiettivo dell'addestramento è quello di minimizzare le cosiddette *loss function*, funzioni di perdita, che misurano la differenza fra le predizioni del modello e i valori effettivamente calcolati nella fase d'addestramento; in

[7]Multi-Scale Short-Time Fourier Transform.

questo caso sono presenti tre categorie di perdite: perdita di ricostruzione (*reconstruction loss*), perdita percettiva (*perceptual loss*) e perdita d'impegno dovuta alla RVQ (*RVQ commitment loss*).

Reconstruction Loss Term

La perdita di ricostruzione presenta due termini: uno calcolato nel dominio del tempo e l'altro nel dominio delle frequenze.

Nel dominio del tempo la funzione di perdita è calcolata come errore assoluto medio (MAE):

$$\ell_t(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_1 \tag{3.1}$$

dove $\boldsymbol{x}$ è l'audio target originale e $\hat{\boldsymbol{x}}$ è il segnale compresso. Per $\|\cdot\|_1$ si intende la norma L^1.

Per quanto riguarda la componente del dominio delle frequenze, si utilizza una combinazione lineare fra le perdite in L^1 e L^2, calcolate avvalendosi degli spettogrammi di Mel di $\boldsymbol{x}$ e $\hat{\boldsymbol{x}}$. Formalmente:

$$\ell_s(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{|\alpha| \cdot |s|} \sum_{\alpha_i \in \alpha} \sum_{i \in e} \|\mathcal{S}_i(\boldsymbol{x}) - \mathcal{S}_i(\hat{\boldsymbol{x}})\|_1 + \alpha_i \|\mathcal{S}_i(\boldsymbol{x}) - \mathcal{S}_i(\hat{\boldsymbol{x}})\|_2 \tag{3.2}$$

dove $\mathcal{S}_i$ rappresenta uno spettogramma di Mel a 64 bin ottenuto tramite STFT normalizzata, e appartiene ad un insieme di valori E che permettono di regolare i parametri del modello e α e s sono dei coefficienti che permettono la normalizzazione del risultato. Per $\|\cdot\|_2$ si intende la norma L^2.

Discriminative Loss

La perdita percettiva è dovuta alla presenza di alcuni discriminatori, inseriti nel modello, per poter migliorare in fase di addestramento la qualità dei segnali compressi prodotti dal generatore, rappresentato in questo caso dal decoder. Il discriminatore utilizzato è un MS-STFTD (vedi Fig. 3.3), ossia una rete neurale convoluzionale che partendo dalla STFT del segnale compresso generato dal decoder riesce a riconoscere diverse strutture del segnale audio e a stabilire se il segnale analizzato sia stato compresso o meno. Il generatore cerca di produrre segnali con una qualità sempre maggiore in modo tale da allenare il discriminatore a distinguere differenze sempre più sottili fra segnale compresso e segnale originale e grazie al lavoro del discriminatore, a sua volta, il generatore risulterà in grado di produrre campioni migliori.

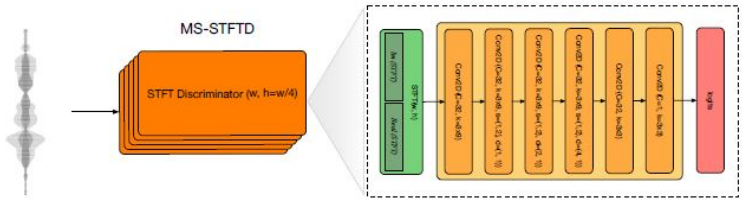

Figura 3.3: Architettura di un discriminatore MS-STFTD [4].

La funzione di perdita del decoder è la seguente:

$$\ell_g(\hat{\boldsymbol{x}}) = \frac{1}{K}\sum_k max(0,1 - D_k(\hat{\boldsymbol{x}})) \tag{3.3}$$

dove K è il numero di discriminatori.

I discriminatori sono invece allenati per minimizzare la seguente funzione di perdita:

$$L_d(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{K}\sum_{k=1}^{K} max(0,1 - D_k(\hat{\boldsymbol{x}})) + max(0,1 + D_k(\hat{\boldsymbol{x}})) \tag{3.4}$$

Durante l'addestramento, il discriminatore tende a sovrastare il decoder. Per fare in modo che il generatore continui a produrre segnali statisticamente realistici, è necessario aggiornare i pesi del discriminatore a 0.66 per segnali a 24 kHz e a 0.5 per segnali a 48 kHz.

RVQ Commitment Loss

EnCodec viene addestrato in modo tale che per segnali a 24 kHz possa supportare larghezze di banda di $1.5, 3, 6$ e 12 kb/s, mentre per i segnali audio stereo a 48 kHz le larghezze di banda che il modello deve essere in grado di gestire sono $3, 6, 12$ e 24 kb/s. Il *commitment loss* dovuto alla quantizzazione vettoriale residuale è così definito:

$$\ell_w = \sum_{c=1}^{C} \|\boldsymbol{z}_c - q_c(\boldsymbol{z}_c)\|_2^2 \tag{3.5}$$

dove $c \in \{1, ..., C\}$ (con C che dipende dalla larghezza di banda desiderata) è il passo in cui viene calcolato il residuo $\boldsymbol{z}_c$ e $q_c(\boldsymbol{z}_c)$ è il *code-vector* che maggiormente somiglia al residuo.

Se viene utilizzata la codifica entropica, il transformer viene allenato tramite la funzione di perdita ℓ_l.

Capitolo 4

Confronto codec lossy

In questo capitolo è presentato un confronto in termini di prestazioni fra l'algoritmo EnCodec e altri codec di tipo *lossy*. Gli algoritmi presi in considerazione, sono:

- *Opus*, codec adatto sia per il parlato che per suoni naturali, formalizzato come standard dal IETF[1] nel 2012. Nella sua configurazione monofononica supporta velocità di trasmissione pari a 6 kb/s;
- *EVS*, codec sviluppato per VoLTE[2], standardizzato nel 2014 da 3GPP[3]. Supporta bit rate da 5.9 kb/s a 128 kb/s;
- *MP3*, codec approvato come standard da ISO/IEC[4] nel 1993. Supporta diversi bit rate, verrà preso in considerazione a 64 kb/s;
- *Lyra-v2*, algoritmo di compressione neurale per la compressione della voce che funziona a 3.2 kb/s e 6 kb/s;
- *EnCodec*, valutato a diverse larghezze di banda: 1.5 kb/s, 3 kb/s, 6 kb/s, 9 kb/s.

Per esaminare la prestazioni sono state considerate metriche di valutazione sia soggettive che oggettive. Per il test soggettivo è stato seguito il protocollo MUSHRA[5] che viene comunemente utilizzato per analizzare la qualità percettiva degli algoritmi di compressione con perdite di segnali audio. Ai partecipanti del test sono stati proposti casualmente 50 campioni audio di differente natura dei quali hanno dovuto valutare la qualità percepita su una scala da 1 a 100. Fra questi campioni è anche presentato indirettamente il cosiddetto

[1] Internet Engineering Task Force.
[2] Voice over LTE.
[3] 3rd Generation Partnership Project.
[4] International Organization for Standardization/ International Electrotechnical Commission.
[5] Multiple Stimuli with Hidden Reference and Anchor.

ancoraggio nascosto, un segnale a qualità nettamente inferiore rispetto gli altri che viene sfruttato dagli ascoltatori per attribuire i vari punteggi. Al termine del test è stata stilata una tabella con i risultati, nella quale è presente il riferimento nascosto, ossia un segnale audio di alta qualità che viene preso come parametro di confronto implicito per tutti gli altri campioni.

Per quanto riguarda i test di natura oggettiva sono stati adoperati il ViSQOL[6] e il SI-SNR[7]. Il primo calcola un indice di qualità che rappresenta la differenza percettiva fra segnale compresso e segnale originale, il secondo invece misura questa discrepanza mediante l'SNR. Per entrambi, maggiore è il punteggio di queste valutazioni, maggiore è la qualità del segnale compresso a livello di percezione uditiva.

Prendendo in considerazione segnali audio monofonici i risultati sono riportati in Figura 4.1 e nelle Tabelle 4.1 e 4.2.

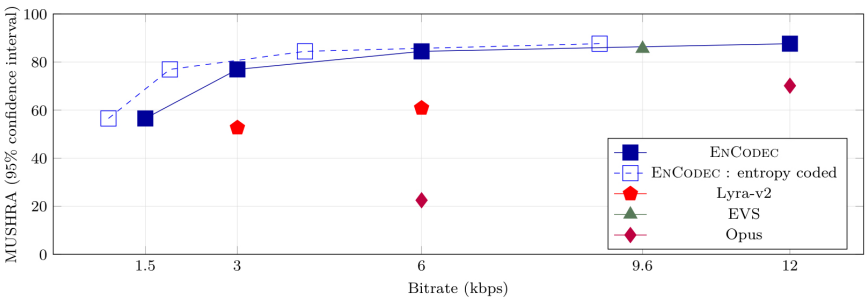

Figura 4.1: Test MUSHRA effettuato su larghezze di banda di differenti codec [4].

Model	Bandwidth	Entropy Coded	Clean Speech	Noisy Speech	Music Set-1	Music Set-2
Reference	-	-	95.5±1.6	93.9±1.8	93.2±2.5	97.1±1.3
Opus	6.0 kbps	-	30.1±2.8	19.1±5.9	20.6±5.8	17.9±5.3
Opus	12.0 kbps	-	76.5±2.3	61.9±2.1	77.8±3.2	65.4±2.7
EVS	9.6 kbps	-	84.4±2.5	80.0±2.4	89.9±2.3	87.7±2.3
Lyra-v2	3.0 kbps	-	53.1±1.9	52.0±4.7	69.3±3.3	42.3±3.5
Lyra-v2	6.0 kbps	-	66.2±2.9	59.9±3.3	75.7±2.6	48.6±2.1
EnCodec	1.5 kbps	0.9 kbps	49.2±2.4	41.3±3.6	68.2±2.2	66.5±2.3
EnCodec	3.0 kbps	1.9 kbps	67.0±1.5	62.5±2.3	89.6±3.1	87.8±2.9
EnCodec	6.0 kbps	4.1 kbps	83.1±2.7	69.4±2.3	92.9±1.8	91.3±2.1
EnCodec	12.0 kbps	8.9 kbps	90.6±2.6	80.1±2.5	91.8±2.5	92.9±1.2

Tabella 4.1: Punteggi MUSHRA per segnali audio compressi di diversa natura campionati a 24 kHz. I risultati sono la media aritmetica dei diversi punteggi ottenuti con intervalli di confidenza del 95%[4].

[6]Visual and Speech Quality Objective Link.
[7]Scale-Invariant Signal-to-Noise Ration.

Model	Streamable	SI-SNR	ViSQOL
Opus	✓	2.45	2.60
EVS	✓	1.89	2.74
EnCodec	✓	6.67	4.35
EnCodec	✗	7.46	4.39

Tabella 4.2: Valutazioni a 6 kb/s di un segnale mono che comprende sia parlato che musica [4].

Nel grafico e nelle tabelle soprastanti è stato dimostrato come EnCodec preservi una qualità percettiva del suono migliore rispetto a tutti gli altri codec quando si considera la stessa larghezza di banda. In Tab. 4.2 si può notare come la configurazione *streamable* di EnCodec presenti un lieve degrado della qualità rispetto alla sua versione *non-streamable*.

Nella Tabella 4.3 sono invece riportati i risultati del test MUSHRA considerando come ingresso dei codec un segnale audio stereo, rendendo questi punteggi molto significativi per quanto riguarda la compressione di file musicali.

Model	Bandwidth	Entropy Coded	Compression	MUSHRA
Reference	-	-	1×	**95.1**±1.8
MP3	64 kbps	-	24×	82.7±3.2
Opus	6 kbps	-	256×	17.7±5.9
Opus	24 kbps	-	64×	82.9±3.7
EnCodec	6 kbps	4.2 kbps	256×	82.9±2.4
EnCodec	12 kbps	8.9 kbps	128×	**88.0**±2.7
EnCodec	24 kbps	19.4 kbps	64×	**87.5**±2.6

Tabella 4.3: Punteggi test MUSHRA [4].

In quest'ultima tabella è mostrato come EnCodec superi nettamente Opus e come sia quasi equivalente a MP3 a 6 kb/s. EnCodec a 12 kb/s raggiunge performance quasi uguali rispetto alla sua versione a 24 kb/s.

Capitolo 5

Conclusione

In questa tesi è stata presentata una panoramica generale sullo stato dell'arte della compressione con perdite di segnali audio. Inizialmente è stato evidenziato come la combinazione fra modello psicoacustico e quantizzazione sia stata la prima tecnica adoperata da diversi algoritmi, come ad esempio MP3 e AAC, per comprimere i segnali audio, gettando le basi per tutti gli studi effettuati successivamente nel mondo della compressione con perdite. In secondo luogo si è dimostrato come l'Intelligenza Artificiale si sia rivelata un efficace strumento anche per il mondo della compressione dei dati. Prendendo in considerazione un algoritmo innovativo, Encodec, che sfrutta le reti neurali per comprimere segnali audio, si è mostrato infatti come l'utilizzo di tecniche tipiche del Deep Learning abbia portato ad un notevole miglioramento delle performance di questo codec rispetto agli algoritmi tradizionali sia per quanto riguarda il tasso di compressione che per la qualità percettiva del suono compresso che all'orecchio umano risulta quasi del tutto indistinguibile rispetto alla sua versione originale.

Nonostante l'impiego dell'AI abbia ottenuto risultati molto soddisfacenti, risulta comunque necessario perfezionare queste tecniche in quanto il problema della gestione dei dati in rete è in continua crescita e bisognerà essere in grado di superare alcune limitazioni come ad esempio il fatto che molte esperienze multimediali, come lo streaming online e le videochiamate, necessitano di una connessione internet veloce e di molto spazio di archiviazione a disposizione. Anche l'emergente mondo del metaverso necessiterà di algoritmi di compressione sempre più sofisticati per potersi espandere. Sarà dunque interessante vedere come saranno raffinate le tecniche già esistenti per rispondere alle nuove esigenze nell'era dei Big Data.

Bibliografia

[1] Z. Abduljabbar e A. Al-Saleh. «A Survey paper on Lossy Audio Compression Methods». In: *Journal of Al-Qadisiyah for Computer Science and Mathematics* 13 (set. 2022), p. 122. doi: 10.29304/jqcm.2021.13.3.853.

[2] Karlheinz Br et al. «Mp3 And Aac Explained». In: (mar. 2001).

[3] D.Pan. «A Tutorial on MPEG/Audio Compression». In: *Readings in Multimedia Computing and Networking*. The Morgan Kaufmann Series in Multimedia Information and Systems. Morgan Kaufmann, 2002, pp. 42–56. isbn: 978-1-55860-651-7. doi: https://doi.org/10.1016/B978-155860651-7/50090-5.

[4] A. Défossez et al. *High Fidelity Neural Audio Compression*. 2022. arXiv: 2210.13438 [eess.AS].

[5] Z. Li, M. S. Drew e J. Liu. *Fundamentals of Multimedia*. Springer, 2021. isbn: 9783030621230.

[6] W. Liu et al. «A High Fidelity and Low Complexity Neural Audio Coding». In: *ArXiv* abs/2310.10992 (2023). url: https://api.semanticscholar.org/CorpusID:264172878.

[7] Inc. Meta Platforms. *Using AI to compress audio files for quick and easy sharing*. 2022. url: https://ai.meta.com/blog/ai-powered-audio-compression-technique/.

[8] M. Zorzi N. Benvenuto. *Principles of Communications Networks and Systems*. Wiley, 2011. isbn: 9780470744314.

[9] M. Ramponi. *What is Residual Vector Quantization?* 2023. url: https://www.assemblyai.com/blog/what-is-residual-vector-quantization/.

[10] K. Sayood. *Introduction to Data Compression*. Fourth Edition. Morgan Kaufmann, 2012. isbn: 9780124157965.

[11] B. Nemeth Wikimedia Commons. *Memory needs by wav and MP3*. 2008. url: https://en.wikipedia.org/wiki/Data_compression#/media/File:MP3_compression.jpg.

[12] Y. Yang, S. Mandt e L. Theis. «An Introduction to Neural Data Compression». In: *Foundations and Trends® in Computer Graphics and Vision* 15.2 (2023), pp. 113–200. doi: 10.1561/0600000107.

[13] N. Zeghidour et al. *SoundStream: An End-to-End Neural Audio Codec*. 2021. arXiv: 2107.03312 [cs.SD].