SAPIENZA
UNIVERSITÀ DI ROMA

Neural Style Transfer e Genre Classification di Brani Musicali tramite Spettrogramma

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea in Ingegneria Informatica e Automatica

Candidato

Lorenzo Palaia
Matricola 1864692

Relatore

Thomas Alessandro Ciarfuglia

Anno Accademico 2023/2024

Sommario

L'obiettivo di questo lavoro è quello di esaminare le implementazioni di Neural Style Transfer (NST) e Genre Classification (GC) di brani musicali sfruttando i dati estratti dai relativi spettrogrammi, realizzate nell'ambito del corso "Laboratorio di Intelligenza Artificiale e Grafica Interattiva". Uno spettrogramma è a tutti gli effetti una rappresentazione grafica dell'intensità di un suono in funzione del tempo e della sua frequenza e questo ci è sufficiente per poterne elaborare i relativi dati.

Il NST è una tecnica di Deep Learning che sfrutta una rete neurale convoluzionale per trasferire uno stile artistico da un'opera all'altra, ad esempio applicando uno stile pittorico ad un'immagine. Si tratta di un metodo di apprendimento non supervisionato che consente di applicare le caratteristiche di un certo stile da una fonte ad un'altra, cercando di mantenere l'aspetto generale dell'oggetto originale. Ciò che andremo a fare è traslare questo discorso in ambito musicale passando per gli spettrogrammi.

In aggiunta a questo, verranno valutate le prestazioni della classificazione dei generi di brani musicali basata sugli spettrogrammi estratti. La classificazione è invece un compito che solitamente rientra nel dominio dell'apprendimento supervisionato, basato su dati etichettati. Sono stati presi in considerazione principalmente due approcci, con tecniche differenti, ognuno dei quali è stato testato con diverse architetture di reti neurali.

Le due analisi che andremo a fare sono quindi ben distinte tra di loro, soprattutto per via del processo con cui si arriva al risultato. Il NST segue infatti un approccio generativo, senza passare per una fase di allenamento ma solo attraverso un'ottimizzazione iterativa del risultato, a differenza del GC che invece riesce ad arrivare al risultato imparando a conoscere i dati etichettati e facendo predizioni.

Tratteremo gli strumenti, le metodologie e i processi attuati in fase di progetto. Discuteremo poi di quale modello si è dimostrato il più affidabile nell'elaborazione delle caratteristiche spettrali dei brani musicali per entrambi i compiti e infine analizzeremo i risultati e le possibili implementazioni future.

Indice

1 Introduzione **1**

2 Neural Style Transfer **3**
2.1 Strumenti 4
2.1.1 Dataset 4
2.1.2 Rete Neurale 5
2.2 Metodologia 6
2.3 Risultati 10

3 Genre Classification **15**
3.1 Strumenti 15
3.1.1 Dataset 15
3.1.2 Reti Neurali 16
3.2 Metodologia 19
3.2.1 Dati .csv 19
3.2.2 Data Augmentation 24
3.3 Risultati 26
3.3.1 Allenamento 26
3.3.2 Predizioni 29

4 Conclusioni **30**
4.1 Neural Style Transfer 30
4.2 Genre Classification 30
4.2.1 Dati .csv 30
4.2.2 Data Augmentation 31
4.3 Sviluppi futuri 31

Bibliografia **33**

Capitolo 1

Introduzione

L'apprendimento automatico è una disciplina di ricerca in continua espansione, con diverse applicazioni in svariati campi. Uno di questi è l'analisi e la manipolazione dei segnali audio, su cui sono stati fatti progressi significativi nell'utilizzo di reti neurali negli ultimi anni. Alcune delle applicazioni più interessanti sono il Neural Style Transfer e il Genre Classification di brani musicali tramite spettrogramma.

L'obiettivo principale del progetto è quello di sviluppare un sistema in grado di riconoscere automaticamente i generi musicali di un brano e in più di generare automaticamente una nuova melodia a partire da uno stile musicale preesistente. Entrambe le tecniche si basano su delle reti neurali capaci di apprendere i caratteri distintivi di un brano musicale specifico a partire dal relativo spettrogramma, ovvero i dati estratti da una rappresentazione in tempo-frequenza di un segnale audio.

È chiaro come sistemi del genere al giorno d'oggi possano portare importanti innovazioni nell'ambito musicale e tecnologico. Poter etichettare i brani musicali automaticamente potrebbe ad esempio permettere di generare delle playlist a tema oppure potrebbe essere la base su cui costruire un sistema di raccomandazione di brani musicali a partire da un genere.

Sviluppi basati sul NST potrebbero invece riguardare la sintesi vocale o la creazione di strumenti a supporto dell'industria della produzione musicale. Creare una nuova melodia a partire da un riferimento può sicuramente aiutare un artista nel processo creativo, fornendo degli spunti da cui partire. È una chiara dimostrazione di come l'intelligenza artificiale stia cominciando ad essere sempre di più uno strumento di supporto e di automazione nella società odierna. Allo stesso tempo è però giusto regolamentare l'utilizzo di queste tecnologie, in particolare dell'intelligenza artificiale generativa, e in questo senso già molte figure si stanno muovendo in questa direzione. Un esempio è quello di IMPF, un ente che rappresenta le società di distribuzione musicale indipendenti in tutto il mondo, che ha prodotto un documento [1] contenente le linee guida riguardanti l'etica di utilizzo dell'intelligenza artificiale generativa, il cui fulcro ruota attorno alla necessità di etichettare le opere musicali generate dall'intelligenza artificiale in modo tale da poterle differenziare da quelle prodotte in maniera autonoma dagli artisti.

L'effettiva implementazione è stata eseguita in Python sfruttando le più importanti librerie di elaborazione dati quali Pandas e NumPy. I risultati più significativi sono stati ottenuti utilizzando la libreria TensorFlow per strutturare le varie reti neurali ma verranno discussi anche alcuni approcci che hanno visto l'utilizzo di PyTorch. In realtà capiterà anche di utilizzare Keras, che è un framework distinto da TensorFlow in quanto il primo può essere integrato come interfaccia ad alto livello all'interno del secondo. In più, per lo studio dei dataset e la visualizzazione dei dati le librerie utilizzate sono state principalmente Matplotlib e Seaborn. Per le operazioni relative ai file audio la libreria di riferimento è invece Librosa.

È bene sottolineare che le due tecniche sono indipendenti tra di loro e vengono realizzate da reti neurali distinte che eseguono compiti totalmente differenti. Le motivazioni di tale approccio sono di natura prettamente tecnica e verranno perciò fornite in seguito. In questa relazione si esamineranno in dettaglio i processi di apprendimento automatico che stanno alla base di questo sistema, così come i risultati ottenuti. Infine, si discuteranno le limitazioni attuali e le possibili aree di sviluppo futuro.

Capitolo 2

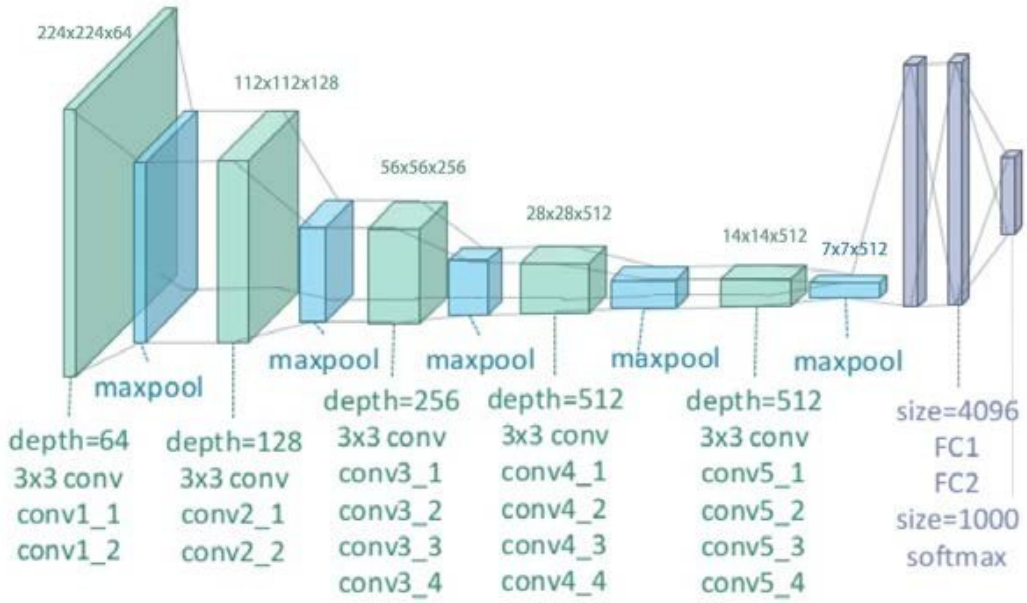
Neural Style Transfer

Il NST è una tecnica di Deep Learning che si concentra sulla manipolazione artistica delle immagini. Introdotta per la prima volta da *Gatys et al.* [2] nel 2015, questa metodologia offre un modo innovativo per trasferire lo stile artistico di un'opera d'arte a un'altra immagine. Si basa sull'utilizzo di reti neurali convoluzionali (CNN), come ad esempio VGG19, che è una rete preaddestrata su un vasto dataset di immagini. Le CNN sono particolarmente adatte al processamento di immagini e si basano su alcuni concetti chiave:

- Convoluzione: la convoluzione consiste nello scorrimento di un filtro (detto kernel) lungo l'intera immagine. Il filtro rileva caratteristiche specifiche come ad esempio bordi o texture, creando delle *mappe di attivazione*.
- Raggruppamento (Pooling): il pooling riduce la dimensione spaziale delle mappe di attivazione preservando le caratteristiche più rilevanti. Il più utilizzato solitamente è il *max pooling*, che seleziona il valore massimo da ogni regione.
- Strati Densamente Connessi (Fully Connected Layers): questi strati aggregano le informazioni estratte dalle fasi precedenti e producono l'output finale. Sono generalmente utilizzati per la classificazione o altre attività specifiche di computer vision. VGG19 è infatti una rete che è normalmente impiegata per la classificazione di immagini.

La struttura della rete VGG19 rispecchia tutte queste caratteristiche ed è un ottimo esempio per introdurre le CNN.

Figura 2.1. Struttura della rete VGG19



L'obiettivo del Neural Style Transfer consiste nell'applicare lo stile di un'immagine ad un'altra cercando però di mantenerne il contenuto intatto. Immaginiamo di avere un'immagine raffigurante un cane e un quadro di Van Gogh, il cui stile artistico è ben riconoscibile. Ci aspettiamo che l'output del NST sia il quadro di un cane dipinto in stile Van Gogh. Ora trasliamo il discorso in ambito musicale: utilizziamo gli spettrogrammi di due tracce per applicare uno stile di un brano ad un altro.

2.1 Strumenti

Come già descritto in precedenza utilizziamo il linguaggio di programmazione Python con la libreria TensorFlow per strutturare il modello della rete neurale che eseguirà il NST. Scomponiamo la nostra analisi degli strumenti in dataset e modelli di reti neurali utilizzati.

2.1.1 Dataset

Il dataset utilizzato nel NST è relativamente piccolo. Si tratta di una serie di tracce audio su cui sono stati effettuati un campionamento ed una successiva conversione in spettrogramma. Ciò è stato possibile sfruttando il modulo Librosa, che permette di eseguire operazioni di campionamento operando quella che si chiama Short-Time Fourier Transform (STFT), che introdurremo in seguito. La relativa limitatezza del dataset è dovuta al fatto che il modello di rete neurale utilizzato per il NST non richiede una vera e propria fase di allenamento, come vedremo successivamente. Il dataset si limita quindi ad essere un raggruppamento eterogeneo di tracce audio su cui poter effettuare vari test di interpolazione al fine di valutarne i risultati.

Le caratteristiche di tutte le tracce audio che compongono il dataset sono le seguenti:

- durata di 10s
- campionamento a 44.1KHz
- canale stereo
- formato .mp3

Il campionamento a 44.1KHz ci indica che la forma d'onda è composta da 44100 punti di campionamento, quindi intuitivamente una frequenza di campionamento maggiore corrisponderà a una qualità dell'audio maggiore per via di uno spazio di campionamento più ampio. Il canale stereo ci indica invece la presenza di stereofonia, che prevede due canali generalmente associati al lato destro e sinistro di ascolto. Il canale mono, al contrario, consiste in un unico flusso sonoro.

2.1.2 Rete Neurale

Le strutture sperimentate per le reti neurali sono state varie ma la scelta definitiva è stata quella di un singolo livello di convoluzione con una funzione di attivazione di tipo ReLU. Questo principalmente perché in questo caso la performance complessiva di reti più profonde si è dimostrata sostanzialmente equivalente a quella di una rete a singolo livello di convoluzione. Eventuali livelli aggiuntivi si sarebbero quindi comportati esclusivamente come ridondanze e perciò è stato scelto di eliminarli. Il discorso riguarda anche la rete VGG19 che sarebbe stata utilizzabile previa modifica, ma che per le stesse motivazioni legate alla ridondanza della maggior parte dei livelli non è stata scelta.

Abbiamo già introdotto brevemente cosa si intende per convoluzione, vediamo ora cosa sono una funzione d'attivazione e più in particolare una ReLU. Una funzione d'attivazione, nell'ambito delle reti neurali, è una funzione matematica che determina l'output di un nodo o di un livello di una rete in base agli input ricevuti. La sua funzione principale è quella di introdurre una non linearità nel modello per renderlo sostanzialmente diverso da un semplice modello di combinazione lineare di pesi e input. Ciò aggiunge flessibilità nell'apprendimento, rendendo la rete capace di apprendere modelli più complessi dai dati in input. A livello pratico le funzioni di attivazione regolano il flusso di informazioni attraverso la rete influenzando quali informazioni sono trasmesse attraverso i nodi e quali sono bloccate. Possiamo immaginarle come quelle funzioni che attivano o disattivano un nodo di elaborazione di una rete neurale. Tra le più note troviamo la *Sigmoid*, la *Tanh*, la *Softmax* e la già citata *ReLU*. La formulazione matematica della ReLU (Rectified Linear Unit) è

$$f(x) = \max(0, x) = \begin{cases} x & \text{se } x \geq 0 \\ 0 & \text{altrimenti} \end{cases}$$

Tra i vantaggi della ReLU ci sono sicuramente la semplicità computazionale, l'introduzione di varianza[1] nel modello, l'invarianza al riscalamento[2] e la prevenzione dal *gradient vanishing*. In reti neurali profonde potrebbero infatti venirsi a creare gradienti molto piccoli durante la retropropagazione ma la ReLU evita questa condizione grazie al suo gradiente costante pari a 1 per valori positivi.

È interessante notare come non sia stata trattata la questione dell'allenamento del modello in fase di sperimentazione. La peculiarità del NST è proprio quella di non richiedere un addestramento della rete neurale. Infatti le distanze tra *target features* e *content/style features*, che introdurremo a breve, sono calcolate iterativamente attraverso la tecnica di discesa del gradiente.

2.2 Metodologia

La fase di sperimentazione per il NST ha ricalcato a grandi linee il processo con cui si effettua un generico NST tra immagini. I punti focali consistono principalmente nel trasformare correttamente una traccia audio nel relativo spettrogramma e nello sviluppare una rete neurale performante e in grado di realizzare un NST. I dati elaborati e ritornati dalla rete neurale dovranno poi essere riportati nuovamente sotto forma di traccia audio.

Il processo richiede una cura dettagliata di ogni passo. Bisogna assicurarsi che il campionamento delle tracce audio venga eseguito nel miglior modo possibile, in modo da mantenere più informazioni possibile. Allo stesso modo, la rete neurale deve essere realizzata in modo da mantenere un'elevata fedeltà rispetto alle fonti elaborate ed evitando allo stesso tempo di produrre un eccessivo rumore in uscita.

Il primo passo è quello di trasformare le tracce audio in spettrogrammi applicando la STFT alle tracce audio, con una finestra temporale di dimensione 2048. La trasformata di Fourier a breve termine è sostanzialmente una funzione che viene moltiplicata per il segnale audio nel tempo per produrre dei segmenti locali di segnale (finestre) e ha il vantaggio di produrre informazioni più dettagliate sullo spettro del segnale rispetto a una trasformata di Fourier completa sull'intero segnale. La scelta della finestra può influenzare l'efficienza e la precisione della STFT. Volendo formalizzare questo processo possiamo dire che la STFT è una funzione

$$\mathbf{STFT}\{x(t)\}(\tau,\omega) \equiv X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)\cdot w(t-\tau)\cdot e^{-i\omega t}\,dt$$

dove:

- $x(t)$ è il segnale in input
- $w(t-\tau)$ è la finestra applicata al segnale. Più precisamente questa è una funzione limitata nel tempo che è centata attorno a τ
- $e^{-i\omega t}$ rappresenta la parte complessa della funzione esponenziale che contribuisce alla rappresentazione in frequenza

[1] In una rete inizializzata in maniera casuale e con output non nulli il valore atteso di neuroni che si attivano è pari al 50% del totale

[2] Si può dimostrare che $\max(0, ax) = a\max(0, x)$ per $a \geq 0$

Questa è la formalizzazione della STFT a tempo continuo. In realtà il modulo Librosa, di cui sfruttiamo la funzione che esegue la STFT, prevede l'utilizzo della trasformata a tempo discreto [3]:

$$\mathbf{STFT}\{x[n]\}(m,\omega) \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-i\omega n}$$

A questo punto è possibile ottenere facilmente le informazioni di modulo e fase del segnale applicando delle semplici funzioni della libreria Numpy. In questa fase si è scelto di calcolare il logaritmo naturale di $1+x$ del modulo della trasformata di Fourier. Tale scelta previene eventuali problemi numerici di rappresentazione del contenuto spettrale per valori di modulo vicini allo zero, in quanto il logaritmo di zero è indefinito. Facciamo notare che solamente le informazioni sul modulo saranno utili nella realizzazione del NST, mentre la fase andrà ricostruita tramite algoritmi specifici. In questa fase è importante inoltre assicurarsi che il numero di campioni del segnale per le due tracce corrisponda per non incorrere in problemi di *overflow* quando andremo ad eseguire la convoluzione. Lo facciamo tramite un'operazione di *slicing* degli array che contengono i dati spettrali appena ricavati con lo scopo di pareggiarne la lunghezza.

Siamo ora in grado di applicare il NST. Il punto di partenza è quello di un generico NST fra immagini [4, 5]. Come precedentemente discusso è possibile ad esempio applicare uno stile pittorico di un'immagine che indichiamo con a ad un'immagine che definiamo b. Il risultato sarà un'immagine che manterrà il contenuto dell'immagine di b con lo stile applicato dell'immagine a.

Figura 2.2. Esempio di NST con immagine di contenuto e immagine di stile

L'idea è allora quella di applicare questa tecnica agli spettrogrammi che abbiamo ricavato. Sfruttiamo quindi le caratteristiche delle reti convoluzionali che abbiamo introdotto precedentemente.

I risultati delle convoluzioni per ogni livello l sono memorizzati in una matrice $F^l \in \mathbb{R}^{N_l \times M_l}$ dove N_l è il numero di filtri, M_l è la dimensione della mappa delle features e F^l_{ij} è l'attivazione dell'i-esimo filtro in posizione j nel livello l. Per avere una rappresentazione dell'immagine nei diversi strati della gerarchia della CNN si passa alla ricostruzione del contenuto attraverso una discesa di gradiente, partendo da un'immagine di rumore bianco fino ad ottenere i risultati di un determinato strato della CNN simili a quelle delle immagini originali.

La perdita di contenuto (Content Loss) è definita come somma dei quadrati degli errori tra le risposte del filtro dell'immagine originale e quelle dell'immagine generata:

$$\mathbf{L}_{\text{content}}(b, x, l) = \frac{1}{2} \sum_{i,j} (F_{l,ij} - B_{l,ij})^2$$

La derivata della Content Loss rispetto alle attivazioni nello strato l viene poi utilizzata per aggiornare l'immagine generata tramite retropropagazione dell'errore.

Per ogni strato della CNN si costruisce in aggiunta un elemento di valutazione della correlazione delle features dato dalla matrice di Gram [6] $G^l \in \mathbb{R}^{N_l \times M_l}$ dove

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$$

è il prodotto tra le feature map i e j nel livello l. La perdita di stile (Style Loss) è definita come la somma dei quadrati delle differenze tra le matrici di Gram dell'immagine originale e dell'immagine generata:

$$E^l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{ij} - A^l_{ij})^2$$

La perdita di stile complessiva è data dalla somma pesata delle perdite di stile su tutti gli strati considerati:

$$L_{style}(a, x) = \sum_{l=0}^{L} w_l E_l$$

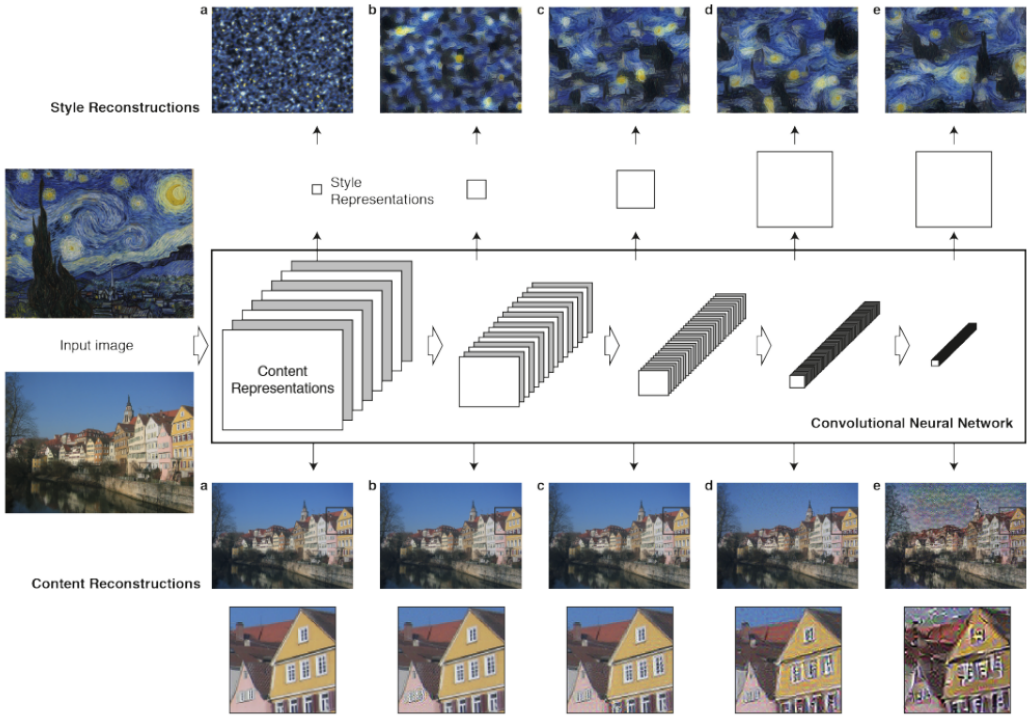
dove w_l sono proprio i pesi di ogni livello che contribuiscono alla perdita di stile totale.

Per arrivare a generare un risultato che unisca il contenuto di b con lo stile di a si deve minimizzare la funzione di perdita totale che è una combinazione lineare delle perdite di contenuto e di stile con i rispettivi pesi α e β:

$$L_{total}(b, a, x) = \alpha L_{content}(b, x) + \beta L_{style}(a, x)$$

α e β sono dei parametri cardine del processo perché andranno stabilire quanto il risultato andrà a mantenere da entrambe le fonti di contenuto e stile.

Figura 2.3. Processo di ricostruzione di stile e contenuto



A questo punto siamo a tutti gli effetti in possesso del risultato finale, uno spettrogramma che dovrà solamente essere riconvertito in segnale audio. Ricordandoci della precedente applicazione di $\ln(1+x)$ sul modulo, procediamo innanzitutto nel riportarlo nelle dimensioni iniziali applicando l'inversa $\exp(x)-1$. Successivamente utilizziamo l'algoritmo di Griffin-Lim [7] per la ricostruzione della fase.

Algoritmo Griffin-Lim

Fissa la fase iniziale $\angle c_0$
Inizializza $c_0 = s \cdot e^{i\angle c_0}$
Itera per $n = 1, 2, ...$
$c_n = P_{C_1}(P_{C_2}(c_{n-1}))$
Fino a convergenza
$x^* = Gc_n$

L'algoritmo si basa sull'iterazione della STFT e della sua inversa (iSTFT) attraverso la proiezione metrica dello spettrogramma su due insiemi C_1 e C_2 con particolari caratteristiche. Questa operazione viene realizzata precisamente dalle funzioni P_{C_1} e P_{C_2}. La formulazione ricorda molto quella di un problema di ottimizzazione e in effetti la convergenza si raggiunge per

$$\min_X ||X - P_{C_1}||_F^2$$

dove $X \in C_2$ e $||\cdot||_F$ è la norma di Frobenius [8] definita come

$$||A||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

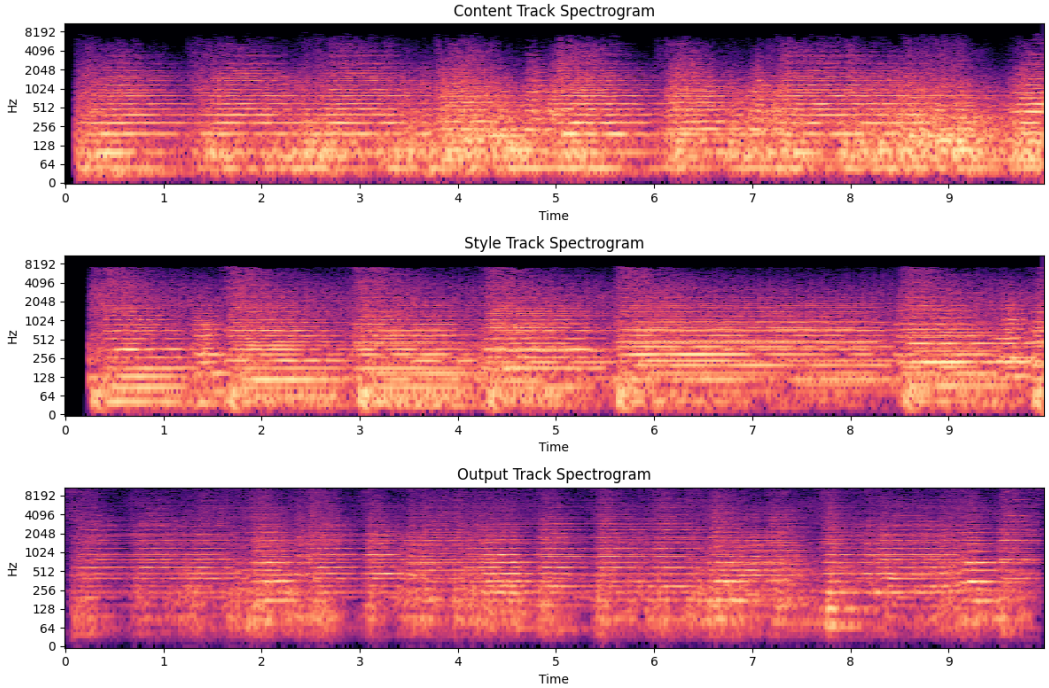
Anche in questo caso ci appoggiamo alle funzionalità del modulo Librosa di Python che integra una versione veloce[3] [9] dell'algoritmo Griffin-Lim che sfrutta un parametro chiamato *momentum* per velocizzare il tempo di convergenza. Chiaramente così facendo stiamo ricostruendo la fase attraverso una stima basata su un algoritmo di minimizzazione e per questo motivo possiamo aspettarci del rumore sulla traccia di output. Per limitarlo a livello puramente sonoro e renderlo trascurabile per le nostre future analisi ci avvaliamo di 500 iterazioni dell'algoritmo. Il processo si conclude quindi con la generazione di una nuova fonte audio con lo stesso campionamento delle tracce in input e, in questo caso, con un canale mono invece che stereo. Abbiamo perciò generato un unico flusso sonoro nonostante gli input fossero stereofonici.

2.3 Risultati

La valutazione di un modello di NST dipende fortemente dalle tracce audio date in input ed è quindi difficile fare delle valutazioni analitiche tali da generalizzare il comportamento del modello indipendentemente dalle fonti audio di riferimento.

Scegliamo di analizzare un caso specifico in cui la traccia audio di contenuto è la *Marcia Imperiale di Star Wars* e la traccia di stile è *The Star-Spangled Banner*, l'inno nazionale degli Stati Uniti d'America. Queste tracce appartengono al dataset introdotto in precedenza e ne rispecchiano le caratteristiche in termini di durata, campionamento, stereofonia e formato. Cominciamo quindi col vedere una rappresentazione grafica degli spettrogrammi di partenza e di output.

Figura 2.4. Rappresentazione degli spettrogrammi di contenuto, stile e output



[3] Per approfondire il Fast Griffin-Lim: https://ltfat.org/notes/ltfatnote021.pdf

A livello puramente visivo, cominciamo col notare come tutte le frequenze della traccia di output siano state attenuate in termini di intensità. Le tonalità gialle, che rappresentano intensità maggiori, scompaiono quasi completamente lasciando posto a intensità più basse e questo comportamento si riscontra principalmente sulle basse frequenze. Quello che notiamo invece per le alte frequenze è che il grafico in output si va a popolare di frequenze che nelle tracce di input sono meno intense o addirittura del tutto assenti. Inoltre, si possono facilmente notare dei salti di intensità delle frequenze che creano delle linee verticali che dividono in segmenti gli spettrogrammi. Queste zone di demarcazione possono indicare variazioni di tono nel brano, cambi di strumento o altre modifiche importanti a livello di frequenza ed è facilmente visibile come alcune di queste zone permangono nello spettro in output.

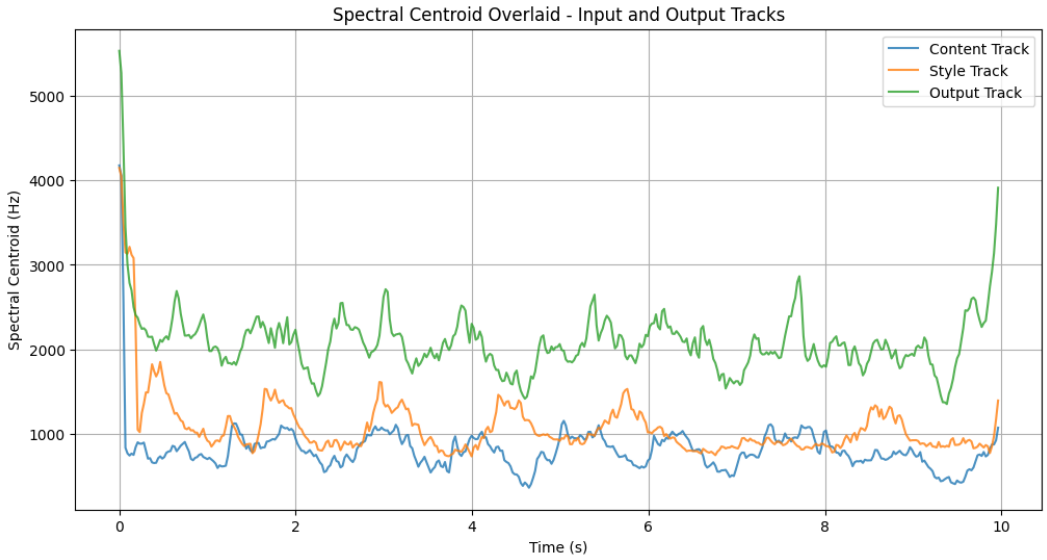
Oltre allo spettrogramma, le metriche su cui confrontare la traccia di output con quelle di input sono molteplici. Partiamo considerando la chiave e il tempo. La chiave di un brano musicale indica la tonalità di base del brano e determina le note e gli accordi che verranno utilizzati. Per indicare le note ci avvaliamo della notazione letterale, in modo da rimanere coerenti con le rappresentazioni che vedremo successivamente. Per la traccia di contenuto sappiamo che siamo in chiave di *G minore* mentre la traccia di stile è in *B bemolle maggiore*. Il brano risultante eredita in questo caso la chiave dalla traccia di stile. Per il tempo accade invece il contrario. Infatti, la traccia di contenuto suona a 103 BPM (beats per minute), quella di stile a 89 BPM e la risultante ha lo stesso tempo della traccia di contenuto.

Se cominciamo poi a entrare più nello specifico attraverso i dati che possiamo estrapolare tramite Librosa, riusciamo a trattare anche metriche più complesse. Introduciamo la centroide spettrale, che è una caratteristica che descrive la distribuzione del suono in una traccia audio. La centroide spettrale è un valore medio che mostra dove si concentra la maggior parte dell'energia sonora in una traccia audio. Questo può essere un utile strumento per determinare se una traccia audio è più acuta o più grave. Una centroide spettrale più alta indica che l'audio è acuto, mentre una centroide spettrale più bassa indica che l'audio è grave. A livello pratico è calcolabile per ogni istante temporale come

$$C = \frac{\sum_1^N fM[f]}{\sum_1^N M[f]}$$

e intuitivamente possiamo interpretarla dalla formula come il centro di massa delle frequenze, in quanto media di tutte le frequenze pesata sulle relative intensità.

Figura 2.5. Analisi della centroide spettrale



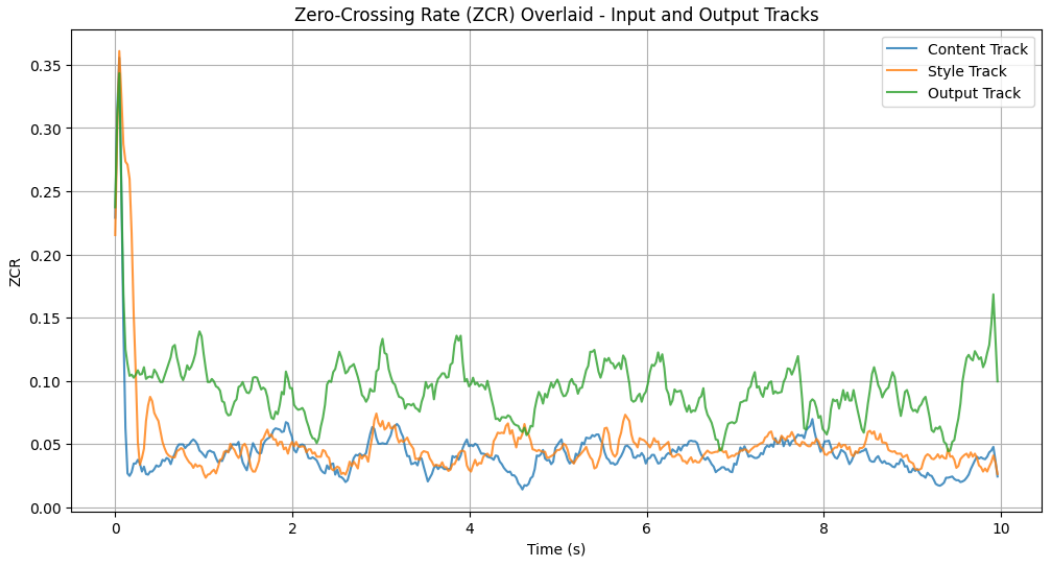
A livello visivo è facile rinvenire molte similitudini tra la centroide spettrale delle tracce di input e quella di output. Si nota come questo valore nel tempo si muova seguendo tendenzialmente lo stesso percorso di una o dell'altra traccia di input senza discostarsi mai bruscamente. Sostanzialmente i picchi e le valli in output sono quasi sempre correlati ad andamenti analoghi nelle tracce di input. Il fatto di ritrovare il grafico dell'output riscalato è un chiaro segno di come le frequenze delle due tracce si siano mescolate fra di loro andando ad innalzare complessivamente il valore del centro di massa delle frequenze.

Possiamo aspettarci lo stesso comportamento per la metrica di Spectral Spread, che non è altro che la deviazione standard rispetto alla centroide spettrale e che quindi ci indica come sono distribuite le frequenze attorno al centro di massa. Un valore basso ci segnala una forte concentrazione di frequenze attorno alla centroide spettrale, mentre un valore alto ci indica una distribuzione delle frequenze più eterogenea e distanziata.

Un'altra nota da menzionare è il fatto che la trasformazione non ha mai generato punti in cui il volume complessivo della traccia supera il valore di 0 dB, considerato la soglia oltre cui si verifica il fenomeno del *clipping*, ovvero il taglio di un'onda sonora per via della sua ampiezza maggiore rispetto all'intervallo in cui è possibile rappresentarla in un sistema digitale.

Possiamo proseguire con l'analisi del rumore introducendo lo Zero Crossing Rate (ZCR), che rappresenta il tasso di variazione di segno del segnale in una determinata finestra temporale. Questo indicatore ci fornisce una misura della rumorosità di un segnale in quanto a un'elevata presenza di rumore corrispondono valori elevati di ZCR.

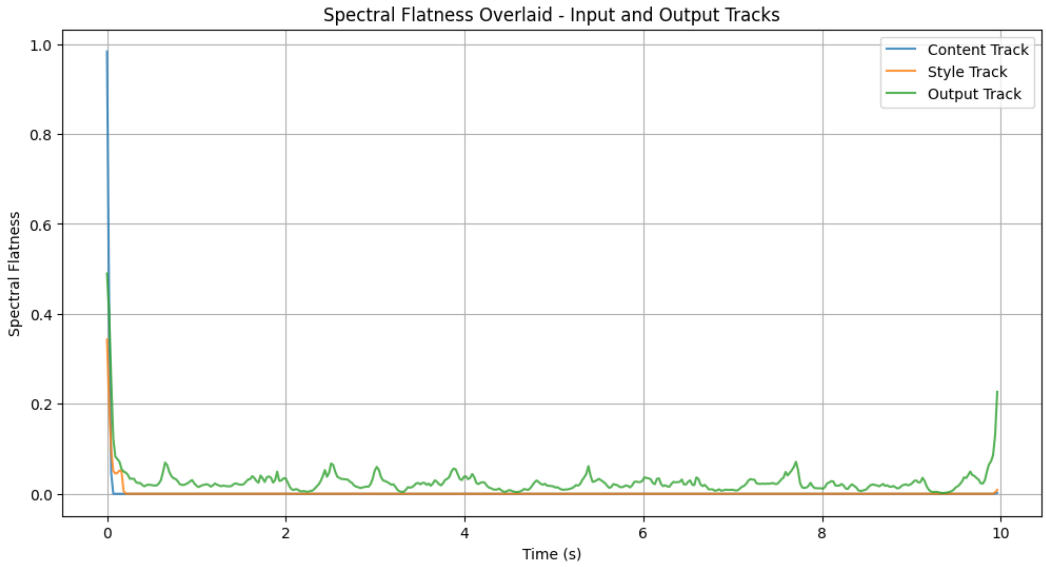
Figura 2.6. Analisi dello ZCR



Anche in questo caso il grafico conferma ciò che abbiamo già notato in precedenza. Il valore di ZCR nell'output non è mai inferiore a quello di nessuna delle due tracce in input, dimostrando una presenza sicuramente maggiore di rumore nell'output.

Discutiamo infine della Spectral Flatness, che fornisce una misura di quanto un suono assomigli a un tono puro piuttosto che a del rumore. In sostanza ci indica la quantità di picchi o strutture risonanti in uno spettro di potenza, considerando che lo spettro di un rumore bianco è completamente piatto. E in effetti un valore di flatness vicina al valore 1 indica che lo spettro ha una quantità simile di potenza in tutte le bande spettrali, che è il comportamento classico del rumore bianco. Al contrario un valore tendente allo zero ci dice che la potenza è concentrata in un numero relativamente piccolo di bande e il relativo segnale suona come una composizione di più onde sinusoidali. Questo valore è calcolabile come il rapporto tra la media geometrica e la media aritmetica dello spettro di potenza.

Figura 2.7. Analisi della Spectral Flatness



È facilmente visibile come i valori di flatness delle tracce in input si attestino sempre intorno allo zero mentre la traccia d'uscita mostra vari picchi al di sopra dello zero. Ci sono quindi dei punti in cui è obiettivamente presente del rumore anche se in maniera molto contenuta.

Mettendo insieme le considerazioni fatte sui tre grafici capiamo come il NST abbia influito sulle tracce di input. Riassumendo, il centro di massa delle frequenze si è spostato più in alto per via della composizione di frequenze diverse mentre il rumore in uscita, seppur molto lieve, è dovuto principalmente alla ricostruzione della fase tramite l'algoritmo di Griffin-Lim in quanto le tracce di partenza presentano valori di rumore tendenzialmente trascurabili. Il valore basso che notiamo nella flatness dell'output ci assicura però che il contenuto è prevalentemente di tipo armonico e non rumoroso.

Un elemento ricorrente in tutti i grafici è la presenza di picchi nettamente elevati all'inizio della tracce. Anche in questo caso la flatness ci viene in aiuto. L'esistenza di un picco iniziale ci sta sostanzialmente indicando la presenza di rumore bianco all'inizio delle tracce, seppur non udibile per via della durata quasi istantanea. Il rumore bianco è un particolare tipo di rumore caratterizzato dall'insieme di tutti i toni possibili nello spettro sonoro, aventi lo stesso livello di ampiezza, ma senza la periodicità nel tempo. Allora riusciamo anche a giustificare il picco iniziale nella centroide spettrale che si presenta estremamente elevata per via della presenza di ogni frequenza dello spettro.

Capitolo 3

Genre Classification

Per il Genre Classification ampliamo il nostro set di strumenti andando a introdurre un nuovo dataset e nuove librerie Python. Per la gestione dei dati tabulari scegliamo di utilizzare Pandas, la libreria di riferimento per la gestione di dati in forma *.csv* (comma separated values). Introduciamo anche la libreria PyTorch per la costruzione di modelli di reti neurali al pari di TensorFlow e di ulteriori moduli aggiuntivi come Torchvision per l'elaborazione di immagini.

3.1 Strumenti

Per la fase di sperimentazione del GC sono stati seguiti due approcci caratterizzati da tecniche radicalmente diverse tra di loro. Entrambe sono poi state testate sviluppando diverse strutture di reti neurali e utilizzando diversi parametri di configurazione.

Le tecniche implementate dipendono strettamente dagli elementi utilizzati all'interno del dataset. In particolare, le rappresentazioni fornite dal dataset per i vari brani musicali sono organizzate sotto forma di file .csv, di immagini rappresentanti gli spettrogrammi e infine di tracce audio. Da qui è stato possibile sviluppare due metodi diversi di allenamento della rete neurale: un primo approccio che sfrutta proprio le features già organizzate nei file .csv ed un secondo che utilizza la tecnica di Data Augmentation, tramite la quale è possibile ampliare i dataset di immagini degli spettrogrammi e di fonti audio.

3.1.1 Dataset

Il dataset su cui andiamo a lavorare è il GTZAN Dataset, uno dei principali dataset per applicazioni di intelligenza artificiale legate alla musica. Le fonti audio utilizzate per costruirlo hanno le seguenti caratteristiche:

- durata di 30s
- campionamento a 22.05KHz
- canale mono
- formato .wav

Facciamo notare come i file in formato .wav godono di una qualità generalmente migliore rispetto a quelli .mp3 per via dell'assenza di compressioni.

Le features contenute nei file .csv sono circa 120 e sono calcolate in due condizioni differenti:

- per l'intera durata delle tracce audio
- dividendo la singola traccia audio in 10 finestre da 3s l'una

Notiamo come il secondo approccio di estrazione delle features consente di avere un dataset 10 volte più ampio rispetto al primo, con evidenti benefici sulla precisione. I modelli sono stati allenati utilizzando il numero maggiore di features possibili. Non sono stati trattati casi di training con sottoinsiemi di features, quindi non è da escludersi l'eventuale presenza di features ridondanti che non migliorano la performance della classificazione. In questo senso è stata però realizzata in fase di sperimentazione una *correlation heatmap* in grado di mostrare quali features sono correlate tra di loro.

Oltre alle tracce audio, gli elementi del dataset utilizzati per fare Data Augmentation sono invece i relativi spettrogrammi in formato di immagine. Le caratteristiche delle immagini sono le seguenti:

- larghezza di 432px
- altezza di 288px
- spazio di colore RGB
- formato .png

Come per il formato .wav, anche il formato .png garantisce un'elevata qualità. Utilizza infatti una compressione senza perdita di dati, il che ci permette di catturare al meglio tutte le informazioni dello spettrogramma.

3.1.2 Reti Neurali

Dati .csv

I modelli testati sono 4 e hanno tutti la struttura di una rete con livelli fully connected (indicata in TensorFlow con layer di tipo Dense). La prima rete testata è costituita esclusivamente da livelli di tipo Dense. A partire dalla seconda rete si è scelto di aggiungere dei layer di *dropout* con percentuale del 20%, ad esclusione del livello di output che ha il compito di stabilire il genere. I livelli di dropout disattivano casualmente un neurone con un tasso del 20% ad ogni iterazione in modo da contrastare l'*overfitting*, che è un problema di cui può soffrire un modello e che approfondiremo in seguito. L'intento è quello di evitare un'eccessiva dipendenza da un determinato neurone a da una specifica feature. La seconda e la terza rete sono strutturalmente identiche ma differiscono nella scelta dell'ottimizzatore: *Adam* per la seconda, *SGD* per la terza. Infine la quarta rete è stata progettata leggermente più profonda, con un tasso di dropout del 30% e *RMSprop* come ottimizzatore.

Un ottimizzatore è sostanzialmente un algoritmo utilizzato per minimizzare la funzione di perdita (loss function) durante il processo di addestramento di un modello. La loss function misura la discrepanza tra le previsioni del modello e i valori effettivi dei dati di addestramento. L'obiettivo dell'ottimizzazione è regolare i pesi e i bias del modello in modo che la funzione di perdita sia ridotta al minimo, consentendo al modello di fare previsioni più accurate su nuovi dati. In particolare:

- Adam (Adaptive Moment Estimation): adatta dinamicamente i tassi di apprendimento (learning rate) durante l'addestramento, tenendo conto dell'andamento nel tempo dei gradienti
- SGD (Stochastic Gradient Descent): aggiorna i pesi del modello in base al gradiente calcolato su un sottoinsieme casuale dei dati di addestramento ad ogni iterazione
- RMSprop (Root Mean Square Propagation): adatta i tassi di apprendimento in base alla media mobile del quadrato dei gradienti

Le strutture complessive sono riassunte in seguito.

Figura 3.1. Struttura delle reti neurali per GC tramite dati .csv

Model 1

Type	Activation	Output Shape	Params
Dense	ReLU	256	14848
Dense	ReLU	128	32896
Dense	ReLU	64	8256
Dense	Softmax	10	650

Total params: 56,650
Optimizer: Adam

Model 2

Type	Activation	Output Shape	Params
Dense	ReLU	512	29696
Dropout (20%)			
Dense	ReLU	256	131328
Dropout (20%)			
Dense	ReLU	128	32896
Dropout (20%)			
Dense	ReLU	64	8256
Dropout (20%)			
Dense	Softmax	10	650

Total params: 202,826
Optimizer: Adam

Model 3

Type	Activation	Output Shape	Params
Dense	ReLU	512	29696
Dropout (20%)			
Dense	ReLU	256	131328
Dropout (20%)			
Dense	ReLU	128	32896
Dropout (20%)			
Dense	ReLU	64	8256
Dropout (20%)			
Dense	Softmax	10	650

Total params: 202,826
Optimizer: SGD

Model 4

Type	Activation	Output Shape	Params
Dense	ReLU	1024	59392
Dropout (30%)			
Dense	ReLU	512	524800
Dropout (30%)			
Dense	ReLU	256	131328
Dropout (30%)			
Dense	ReLU	128	32896
Dropout (30%)			
Dense	ReLU	64	8256
Dropout (30%)			
Dense	Softmax	10	650

Total params: 757,322
Optimizer: RMSprop

Per concludere, l'allenamento è stato effettuato utilizzando una *batch size* di 128 elementi ed una loss function nota come Sparse Categorical Cross Entropy. Intuitivamente questa funzione di loss calcola l'errore tra le predizioni e le etichette reali, penalizzando maggiormente le predizioni errate. In particolare, viene calcolata la somma dei logaritmi delle probabilità di predizione corrispondenti alle classi corrette, moltiplicati per -1.

Praticamente:

- Se il modello predice correttamente il genere di un brano la probabilità associata alla classe corretta il valore sarà vicino a 1, e il termine logaritmico contribuirà poco o nulla alla perdita
- Se il modello predice in maniera errata la probabilità associata alla classe corretta il valore sarà più vicino a 0, il che aumenterà notevolmente il termine logaritmico e, di conseguenza, la perdita totale

Data Augmentation

Per fare Data Augmentation invece sono stati implementati due modelli di reti neurali corrispondenti a due scelte differenti: quella di ampliare il dataset di immagini e quella di agire direttamente sulle fonti audio. Distinguiamo queste due scelte con i nomi di Image Augmentation e Audio Augmentation.

Per l'Image Augmentation, realizzata in PyTorch, si è scelto un modello a 5 livelli di fully connected con funzioni di attivazione ReLU su tutti i livelli ad esclusione dell'ultimo che, come già sappiamo, farà da vero e proprio discriminatore per il genere musicale. Per motivi di conformità con i modelli precedenti, i livelli di fully connected sono indicati attraverso il tipo Dense ma è bene far notare come in PyTorch la classe equivalente che realizza questo tipo di livello è chiamata Linear.

Figura 3.2. Struttura della rete neurale per GC tramite Image Augmentation

Model 1			
Type	Activation	Output Shape	Params
Dense	ReLU	1024	274640896
Dense	ReLU	512	524800
Dense	ReLU	128	65664
Dense	ReLU	32	4128
Dense		10	330
Total params: 275,235,818			
Loss Function: Cross Entropy Loss			

La rete neurale scelta per l'Audio Augmentation, di cui si omette la rappresentazione grafica per via dell'elevata lunghezza, è costituita da una prima parte di convoluzione e da una seconda parte di fully connected connesse tra di loro da un livello di *flatten*. Vediamo meglio la struttura:

1. Convoluzione: 4 livelli di dimensione 64, 128, 256 e 512 con funzione d'attivazione ReLU, dove ogni livello è seguito da un dropout al 10% e da un max pooling di dimensione 2×2. Come sappiamo, il max pooling ha il compito di ridurre la dimensione spaziale della nostra rappresentazione mantenendo solo i valori massimi all'interno di ciascuna finestra.
2. Flatten: questo livello serve semplicemente a convertire i dati da una dimensione a un altra; in questo caso lo inseriamo perché successivamente interviene la fully connected che lavora con dimensioni inferiori alla convoluzione.
3. Fully Connected: 3 livelli di dimensione 2048, 1024 e 256 con funzione d'attivazione ReLU seguiti da un livello di output di dimensione 10 con attivazione softmax.

La softmax prende in input dei valori reali e li normalizza in modo tale che la somma di tutte le probabilità sia 1. Così facendo la classe con la probabilità più alta diventa quella predetta dal modello. Per via della sua formulazione, la softmax è spesso utilizzata come ultimo strato di una rete neurale in problemi di classificazione multi-classe, in quanto produce un output interpretabile come probabilità di appartenenza alle diverse classi L'ottimizzatore scelto è Adam e la funzione di perdita usata è la Sparse Categorical Cross Entropy.

Essendo entrambe delle tecniche di supervised learning è stata richiesta anche una fase di allenamento dei modelli attraverso la definizione di etichette di genere.

3.2 Metodologia

Trattiamo ora le due metodologie con cui si è raggiunto l'obiettivo di classificare i generi musicali.

3.2.1 Dati .csv

Sfruttiamo le features già calcolate e presenti nel dataset in formato .csv. Visto l'elevato numero di features presenti riportiamo e analizziamo soltanto le principali:

- BPM
- Media e varianza di
  - Ampiezze di banda spettrale
  - Armoniche
  - Chroma Features
  - Rolloff
  - Centroide spettrale
  - ZCR

I BPM sono una misura della velocità di un brano. Indicano quanti battiti ci sono in un minuto di una traccia musicale e può aiutare a determinare se un brano è lento oppure veloce. Un valore di BPM più elevato indica una traccia dal ritmo più veloce, mentre un valore di BPM più basso indica una traccia più lenta.

Le medie e le varianze delle varie features sono calcolate sui file audio del dataset aventi le caratteristiche già discusse in precedenza. Il processo di estrazione di queste features sfrutta la già citata STFT che può essere eseguita in maniera efficiente tramite l'algoritmo di Fast Fourier Transform (FFT). Per ogni finestra temporale presa in analisi sono state calcolate le features a partire da $M[f]$, che è la magnitudine della FFT [10] nell'intervallo di frequenza f.

L'ampiezza di banda rappresenta la differenza tra le frequenze superiori e inferiori in una banda continua di frequenze. Sapendo che i segnali oscillano attorno a dei punti, se consideriamo un punto come il baricentro del segnale, la somma della massima deviazione del segnale in entrambe le direzioni rispetto al punto può essere considerata come la larghezza di banda del segnale in un determinato intervallo temporale. Per esempio, se un segnale presenta frequenze di 1000 Hz, 700 Hz, 300 Hz e 100 Hz, la larghezza di banda del segnale sarà calcolata come la differenza tra la frequenza massima e quella minima, ovvero $1000 - 100 = 900$ Hz.

Le armoniche sono componenti di frequenza di un segnale audio che sono aggiunte intenzionalmente o presenti in maniera naturale. Sono generate da una combinazione di fattori come le onde sonore che vengono riflesse o assorbite dalle pareti, distanze, tipi di superfici e altri fattori presenti nel momento in cui si cattura un'onda sonora. La loro presenza può avere un impatto significativo sul suono di una traccia audio, e possono essere utilizzate per scopi diversi come dare profondità, definizione e corpo alla traccia.

Le chroma features sono un tipo di caratteristica estratta che consente di analizzare una traccia audio in modo più efficiente. Rappresentano una traccia audio in una scala di 12 *semitoni* e consentono di rilevarne le frequenze fondamentali, indipendentemente dalla loro intensità. Il semitono è il più piccolo intervallo musicale tra due suoni e 12 semitoni compongono un'ottava, che possiamo immaginare intuitivamente come i 7 tasti bianchi del pianoforte intervallati dai 5 tasti neri. Tutto ciò consente di misurare la similitudine tra tracce audio in modo più accurato, rilevando armoniche e caratteristiche ritmiche. Una delle peculiarità delle chroma features è il fatto di essere robuste ai cambiamenti di timbro e strumentazione.

Il rolloff [11] è una misura di quanto rapidamente un segnale audio si attenua al di fuori di un intervallo specifico di frequenze. In sostanza descrive l'azione di un tipo specifico di filtro progettato per attenuare progressivamente le frequenze che può essere di tipo *passa-alto* o *passa-basso*. Può essere usato per controllare il suono di una traccia audio, ad esempio attenuando le frequenze al di sopra di una determinata frequenza. Ciò può aiutare a ridurre al minimo l'effetto del rumore, delle interferenze e dei suoni indesiderati nella traccia. Il rolloff può essere regolato in modo da produrre un suono più morbido e naturale o più tagliente e aggressivo. In termini matematici il rolloff si può indicare come quel valore R tale che

$$\sum_{1}^{R} M[f] = 0.85 \sum_{1}^{N} M[f]$$

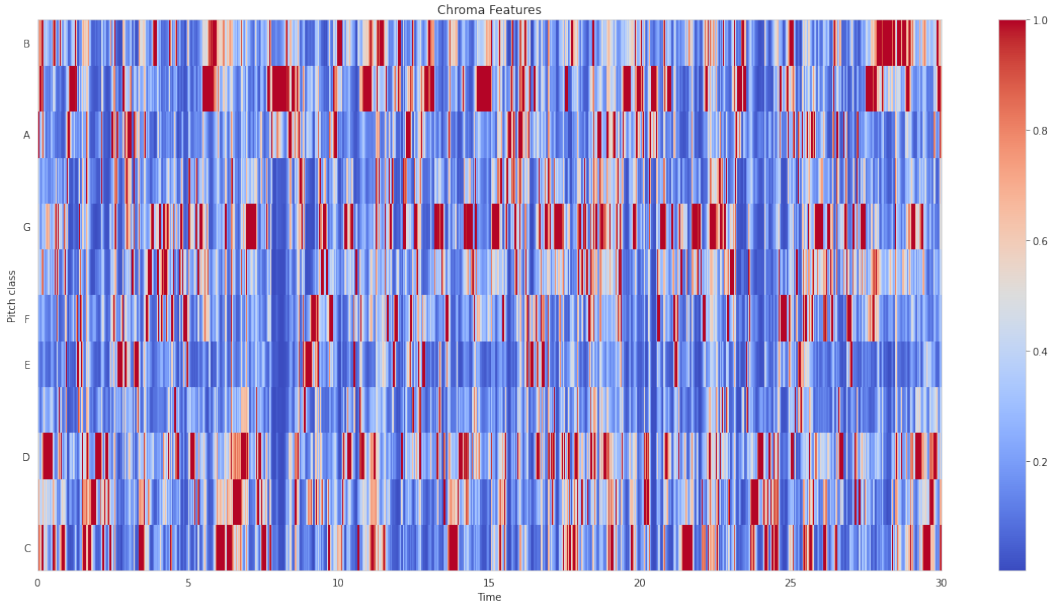
ovvero quello che ci va a considerare la frazione di spettro in cui l'85% (valore nominale) della potenza è a frequenze inferiori. Offre quindi una misura di quanto rapidamente la potenza spettrale diminuisce al di fuori di un intervallo specifico, fornendo informazioni utili sulla distribuzione delle frequenze di un segnale.

La centroide spettrale e lo ZCR sono invece già stati introdotti nella valutazione dei risultati del NST.

Analizzate le features principali, si procede innanzitutto al caricamento del dataset .csv in un oggetto Dataframe della libreria Pandas, tramite cui possiamo avere accesso in futuro a tutte quelle che sono le funzioni di gestione di un dataset fornite da Pandas. A seguire si è reso necessario codificare le etichette di genere attraverso due dizionari che associano in maniera biunivoca i generi a dei valori numerici. L'utilizzo di due dizionari è una scelta dovuta alla necessità di accedere più facilmente a chiavi e valori. Infatti essendoci una corrispondenza biunivoca tra gli elementi è possibile costruire due dizionari in cui le chiavi del primo fanno da valori del secondo e i valori del primo fanno da chiavi nel secondo.

Per studiare meglio i dati possiamo ricavare delle rappresentazioni visuali delle tracce su cui si sta lavorando. Possiamo quindi caricare una traccia audio del dataset ed ottenere le chroma features tramite la funzione *chroma_stft* del modulo Librosa. Il risultato è visualizzabile sotto forma di grafico tempo-tono.

Figura 3.3. Chroma features di una traccia blues

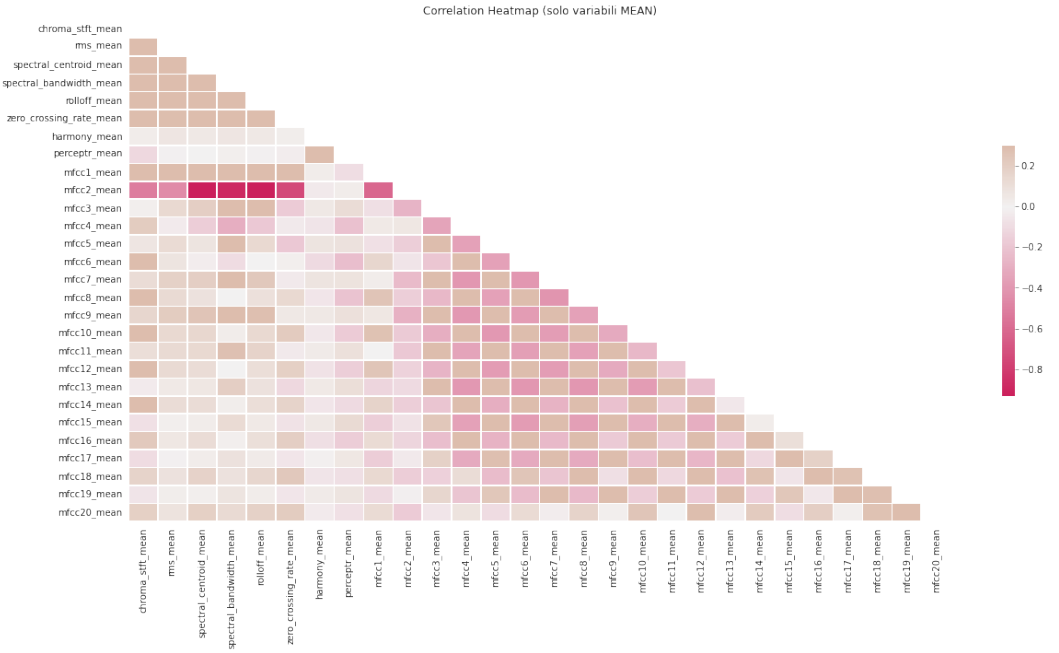


Il grafico ci indica a che al tempo t il semitono s appartenente all'ottava[1] che va da C (Do) a B (Si) è presente con una probabilità p indicata con un'intensità di colore differente. In un certo senso questo grafico può essere associato alla rappresentazione sul pentagramma della traccia audio in questione, anche se non è del tutto completo in quanto. mostrando solamente un'ottava, non è robusto a cambiamenti di quest'ultima in caso una nota venisse suonata più acuta o più grave. È però un ottimo indicatore del tono e della durata di una nota nel tempo.

[1] Sull'asse delle ordinate sono omessi i semitoni diesis e bemolle ma sono comunque visibili nelle bande tra una nota e l'altra

Continuando a visualizzare i dati, possiamo valutare ad esempio quale correlazione intercorre tra le varie features e quali in particolare sono fortemente correlate fra di loro. Ci viene in aiuto la correlation heatmap, realizzabile tramite la libreria Seaborn. Nell'esempio che segue è stata calcolata solamente usando le features *mean*, ovvero le medie calcolate nelle diverse finestre temporali.

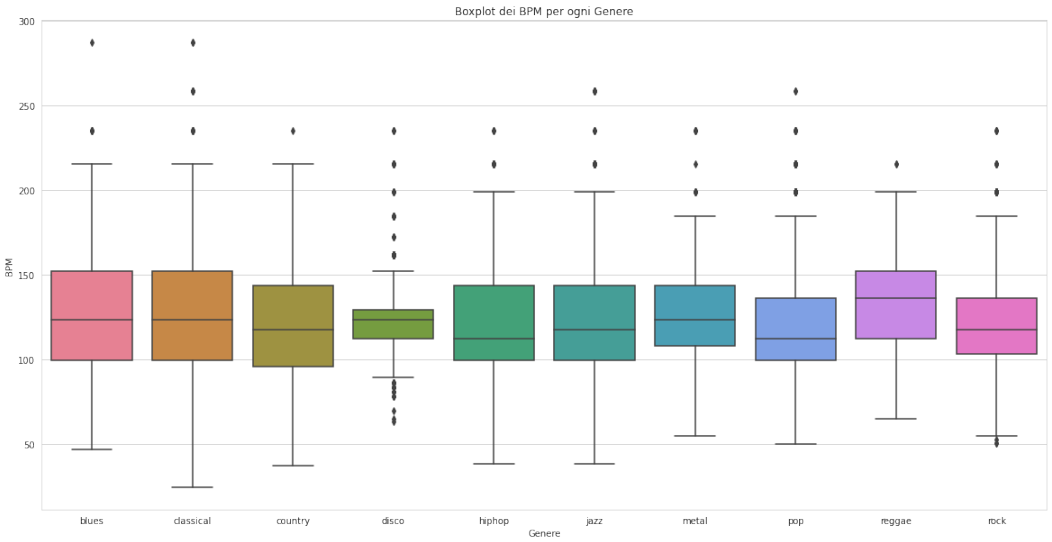
Figura 3.4. Correlation heatmap delle features *mean*



Notiamo come le features sono tendenzialmente non correlate tra di loro anche se c'è un'eccezione per quanto riguarda la variabile *mfcc2_mean*. Infatti questa risulta essere fortemente correlata con le features di centroide spettrale, ampiezza di banda, rolloff e zero crossing rate con un fattore di correlazione che si attesta intorno al -0.8. Ricordando brevemente che un valore positivo del coefficiente di correlazione ci indica che due variabili sono direttamente correlate e che un valore negativo ci indica una correlazione inversa, e considerando che tale coefficiente assume sempre valori compresi tra -1 e 1, deduciamo che la variabile considerata si muove tendenzialmente in maniera opposta rispetto alle altre. In particolare i MFCC (Mel Frequency Cepstral Coefficients) [11] rappresentano l'energia dello spettro di potenza del suono su diverse bande di frequenza, utilizzando una scala di Mel invece della scala lineare di Hertz. Questo perché la percezione umana delle frequenze segue più la prima scala rispetto alla seconda. I MFCC vengono estratti utilizzando una serie di passaggi, tra cui la trasformata di Fourier a breve termine per analizzare lo spettro del suono, la mappatura della scala di Mel per rappresentare l'energia dello spettro su bande di frequenza Mel e la trasformata del coseno discreta per ottenere i coefficienti cepstrali.

Un'ulteriore analisi può essere fatta sugli intervalli di BPM di un genere musicale. Come possiamo immaginarci un genere musicale si attesta solitamente in un intervallo di BPM ben definito. Allora possiamo chiederci quali sono questi intervalli e visualizzarli tramite un boxplot.

Figura 3.5. Boxplot dei BPM per ogni genere



Per ogni genere identifichiamo un rettangolo ed un segmento. Considerando l'insieme dei valori dei BPM di ogni traccia per genere, il rettangolo è delimitato dal primo e dal terzo quartile dell'insieme, $q_{1/4}$ e $q_{3/4}$, e diviso al suo interno dalla mediana $q_{1/2}$. I segmenti sono delimitati dal minimo e dal massimo dell'insieme. In questo modo vengono rappresentati graficamente i quattro intervalli ugualmente popolati delimitati dai quartili. Abbiamo quindi il riscontro visivo del fatto che differenti generi musicali si attestano su differenti BPM.

A seguito della fase di analisi dei dati si può procedere alla preparazione dei dati stessi. Si escludono in maniera preliminare le features che non sono necessarie come ad esempio la lunghezza della fonte audio che, essendo fissata, risulta essere inutile alla classificazione. Costruiamo quindi i tre set di addestramento, validazione e test. In particolare il bilanciamento degli elementi per set risulta essere:

- addestramento: 70%
- validazione: 20%
- test: 10 %

Concludiamo normalizzando le features: centriamo e scaliamo azzerando la media e portando la varianza a 1 tramite la classe StandardScaler della libreria Sklearn.

A questo punto abbiamo visualizzato i dati, li abbiamo preprocessati e siamo pronti ad addestrare il modello.

3.2.2 Data Augmentation

La Data Augmentation è una tecnica utilizzata nell'ambito del machine learning e dell'elaborazione dei dati atto a migliorare le prestazioni dei modelli, specialmente in contesti in cui i dati di addestramento sono limitati o poco eterogenei. Consiste nel creare nuovi dati di addestramento applicando alcune trasformazioni a quelli già esistenti e introducendo variazioni che non ne alterano significativamente il contenuto. Questa tecnica riduce anche il rischio di overfitting, ovvero quella situazione in cui un modello tende ad adattarsi troppo bene ai dati di addestramento mostrando però scarsa flessibilità e capacità di generalizzare su dati nuovi e sconosciuti.

Image Augmentation

Alcune tecniche comuni di Data Augmentation per le immagini coinvolgono le operazioni di rotazione, traslazione, ingrandimento o rimpicciolimento, capovolgimento, aumento o diminuzione di luminosità o contrasto e aggiunta di rumore.

Come realizzato precedentemente con i dati in formato .csv, il primo passo è certamente quello di visualizzare i dati su cui stiamo lavorando. Tramite le librerie PIL e Matplotlib possiamo visualizzare un'immagine del nostro dataset rappresentante uno spettrogramma. Con semplici operazioni riusciamo a risalire alle dimensioni dell'immagine che, come ci aspettiamo, rispecchiano esattamente quelle che abbiamo definito precedentemente nell'introduzione del dataset. Ci viene indicata però una terza dimensione, di valore pari a 4, che rappresenta i canali RGBA[2]. RGBA è uno standard di rappresentazione del colore che introduce il canale *alpha*, il quale rappresenta la trasparenza di un pixel in un intervallo da 0 a 1. È equivalente ad altri standard come CMYK, HSL, HSV o l'esadecimale, ai quali è legato tramite precise relazioni matematiche. In sostanza la nostra immagine è interpretata come la sovrapposizione di 4 canali, rappresentati da matrici di dimensione uguale i cui valori indicano le intensità dei canali rosso, verde, blu e alpha.

Ricordando che anche questo è un metodo di apprendimento supervisionato, abbiamo la necessità di etichettare i dati di addestramento e di codificarli e decodificarli facilmente. Ci sono chiaramente delle librerie di Python che automatizzano questo processo ma è stato scelto di costruire un semplice data encoder/decoder basato su una semplice struttura a dizionario che mappa una determinata etichetta ad un valore intero. Questa tecnica è comunemente chiamata Label Encoding ed è un'alternativa ad altre tecniche come il One-Hot Encoding[3] che è tendenzialmente più dispendioso in termini di memoria.

Si arriva quindi al momento dell'applicazione delle trasformazioni. Tramite le librerie PyTorch e Torchvision applichiamo le trasformazioni che abbiamo trattato in precedenza in maniera casuale, con l'intento di aumentare il più possibile la varianza del dataset.

[2] Da non confondere con lo spazio di colore RGB introdotto in precedenza in quanto concettualmente diversi

[3] Il One-Hot Encoding prevede l'uso di un vettore binario in cui tutti gli elementi sono zero tranne quello relativo alla categoria corrispondente

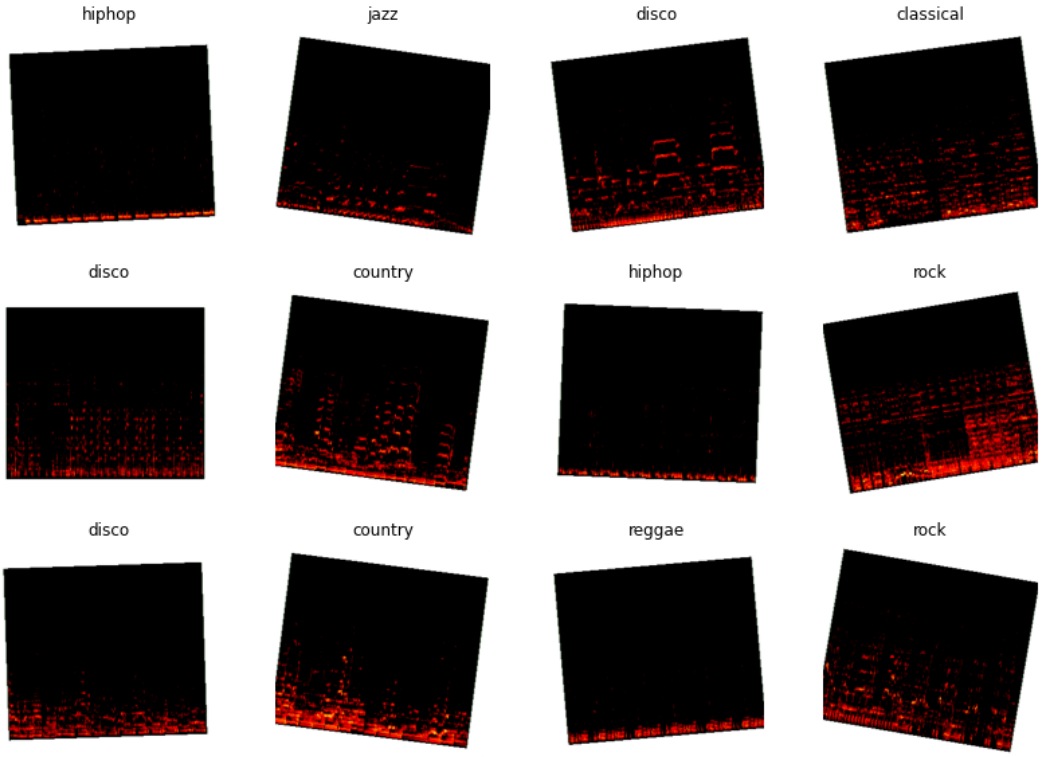
Si procede quindi alla normalizzazione del dataset. L'intento è quello di standardizzare i dati in modo che abbiano una media pari a zero e una deviazione standard unitaria. Stiamo quindi cercando di passare a una distribuzione normale standardizzata. Questo processo è facilmente interpretabile quando si lavora con dati in formato tabulare ma potrebbe essere meno intuitivo in caso di immagini. In maniera semplificata, il dataset viene diviso in batch di 32 elementi, viene effettuata una *reshape* (riformattazione) per riadattare il tensore in una forma che renda possibile il calcolo di questi parametri lungo dimensioni specifiche. Una volta calcolate media e deviazione standard si normalizza sulla base di questi valori e della formula

$$z = \frac{x - \mu}{\sigma}$$

che è la formula della standardizzazione di una variabile aleatoria, nonché la stessa applicata dalla classe StandardScaler di Skleran già citata in precedenza.

Possiamo visualizzare come cambiano le rappresentazioni degli spettrogrammi a seguito delle trasformazioni.

Figura 3.6. Campione di immagini di spettrogrammi dopo l'Image Augmentation



Notiamo come ad esempio le rotazioni applicate alle immagini non sono mai eccessive. In effetti alcune trasformazioni sono state limitate per non perdere eccessivamente informazioni. Applicare una rotazione di 180° ad uno spettrogramma, ad esempio, si tradurrebbe in una completa inversione di tempo e frequenza.

Audio Augmentation

Il processo di Audio Augmentation segue gli stessi concetti introdotti per l'Image Augmentation. La differenza risiede nel fatto che le trasformazioni si vanno ad applicare sulla traccia audio originale e solo successivamente si passa alla rappresentazione spettrale. Questa sarà poi data in input alla rete per la fase di allenamento.

In questo caso decidiamo di applicare principalmente due trasformazioni:

- Aggiunta di rumore gaussiano con ampiezza compresa tra 0.001 e 0.015, con probabilità di applicazione del 70%
- Variazione di tono in un intervallo da -4 a $+12$ semitoni con probabilità di applicazione del 50%

Tutto ciò è reso possibile dalla libreria *audiomentations* che agisce direttamente sui dati della fonte audio.

Si può quindi procedere al calcolo degli spettrogrammi. Utilizziamo le stesse funzioni già trattate in precedenza per il NST tramite Librosa. Questo processo, iterato per le 1000 tracce audio che compongono il dataset GTZAN, può rivelarsi particolarmente dispendioso in termini temporali ma è l'unico modo per andare a fare augmentation sulla fonte audio originale invece che sullo spettrogramma. Per fornire una misura, in un ambiente come quello di Google Colab, l'esecuzione di questa operazione richiede tra i 5 e i 7 minuti.

La preparazione dei dati si conclude con la divisione del dataset in train set all'80% e test set al 20% e il successivo encoding dei generi musicali tramite il LabelEncoder di Sklearn.

3.3 Risultati

Giungiamo quindi al punto di valutare le performance delle implementazioni e dei modelli trattati finora. Andiamo a fare delle considerazioni per quanto riguarda la fase di allenamento e di quella che riguarda le predizioni.

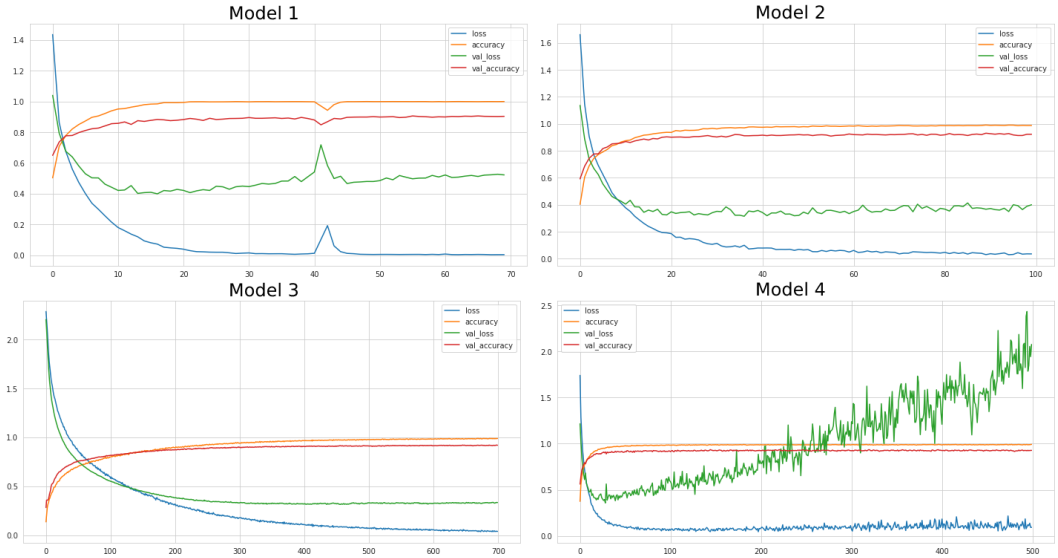
3.3.1 Allenamento

Dati .csv

La fase di allenamento dei modelli è stata eseguita sui 4 modelli con *epoche* differenti. Il concetto di "epoca" si riferisce a una singola iterazione attraverso l'intero set di addestramento. In particolare si è scelto un allenamento in 70 epoche per il primo modello, in 100 epoche per il secondo, in 700 epoche per il terzo e in 500 epoche per il quarto. Gli ottimizzatori scelti sono stati già discussi in precedenza. I risultati che seguono indicano le metriche di *loss* e *accuracy* nei set di allenamento e validazione dove:

- la loss è un valore reale che indica quanto il modello si discosta dalla verità nei suoi output
- l'accuracy è un valore in percentuale del rapporto tra predizioni corrette e numero di elementi di cui predire il genere

Figura 3.7. Andamento del training delle reti per GC tramite dati .csv



La massima precisione nel set di validazione si è attestata nel caso peggiore al 90,54% per il modello 1 e nel caso migliore al 93,57% per il modello 4, nonostante un'impennata nella loss che non si è invece registrata per gli altri modelli. Il training dei modelli 2 e 3 è risultato quello con meno scostamenti in entrambe le metriche di loss e accuracy. Infatti, oltre all'aumento della validation loss nel modello 4, si può notare nel modello 1 come a ridosso dell'epoca 40 ci sia stato un peggioramento di entrambe le metriche che si è comunque immediatamente assesestato sull'andamento asintotico generale.

Riassumiamo i risultati raccolti dall'allenamento in una tabella.

	Train Loss	Train Acc.	Val. Loss	Val, Acc.	Max Val. Acc.
Modello 1	0.0034	99,93%	0.5220	90,29%	90,54%
Modello 2	0.0350	98,80%	0.3996	92,26%	93,07%
Modello 3	0.0391	88,84%	0.3390	91,91%	92,11%
Modello 4	0.0938	99,27%	2.0752	92,82%	93,57%

Tabella 3.1. Risultati di training e validazione per GC tramite dati .csv

Notiamo innanzitutto come il modello che si è dimostrato più preciso sul train set è il modello 1, che dimostra di sbagliare appena 7 predizioni su 10000 nel train set. Nonostante ciò lo stesso modello mostra l'accuratezza minore tra tutti quando si considera il set di validazione. Potrebbe essere un campanello d'allarme che ci indica una possibile presenza di overfitting nel modello, che potrebbe essersi adattato troppo bene al train set senza mostrare eccessiva capacità di generalizzare sui dati. Ciò è confermato anche dai risultati sulla loss, che nel training set si è rivelata essere la più bassa. La validation loss migliore di tutte la fa registrare invece il modello 3, che però allo stesso tempo dimostra di avere la precisione in fase di allenamento minore di tutti. Il modello 2 si dimostra essere il modello più equilibrato, con valori confrontabili con gli altri modelli ma sicuramente più bilanciati.

Il modello 4 ha invece la validation loss maggiore, che è un chiaro segno di overfitting, soprattutto in confronto agli altri risultati. Nonostante ciò questo modello mostra la precisione sul set di validazione maggiore, confermata anche dal parametro Max Validation Accuracy, ed è quello che andremo a valutare in seguito sul test set.

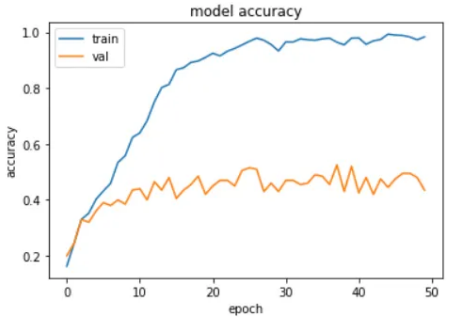
Image Augmentation

Per il modello che sfrutta l'Image Augmentation riusciamo a raggiungere una precisione del $24,05\%$. È un risultato certamente migliorabile ma che richiede comunque uno studio ancor più approfondito dei dati. Bisogna comunque tenere conto del fatto che una tecnica come l'Image Augmentation non è tendenzialmente pensata per essere applicata su degli spettrogrammi. Come abbiamo discusso precedentemente, il set di trasformazioni a nostra disposizione resta relativamente limitato perché rischieremmo di perdere informazioni in maniera troppo eccessiva in caso di utilizzo smisurato delle trasformazioni.

Audio Augmentation

L'allenamento del modello per l'Audio Augmentation ha prodotto il seguente grafico nel tempo.

Figura 3.8. Andamento del training della rete per GC tramite Audio Augmentation



Notiamo come l'andamento del training non sia fluido come nei casi precedenti ma ancor di più come la distanza tra la precisione sul training set e sul validation set sia estremamente marcata, molto più dei modelli trattati in precedenza. Anche in questo caso si sospetta la presenza di overfitting, nonostante le misure prese per prevenirlo.

3.3.2 Predizioni

Dati .csv

Il modello 4 è stato considerato il più performante per via della sua maggiore precisione sul set di validazione. Il passo successivo è stato quello di effettuare delle predizioni sul set di test. La precisione sul test set si attesta al $92,54\%$.

Image Augmentation

Anche nelle predizioni sul test set il modello di Image Augmentation si attesta sui livelli dell'allenamento riuscendo a predire correttamente il genere circa una volta su quattro.

Audio Augmentation

Il modello riesce a raggiungere un'accuratezza del $52,5\%$ sul validation set. Considerando la precisione del modello con Image Augmentation si nota un miglioramento del 25% circa che si riflette in una precisione sostanzialmente raddoppiata.

Capitolo 4

Conclusioni

4.1 Neural Style Transfer

Trarre delle conclusioni per il NST senza avere una misura concreta a livello sonoro dell'output può essere complicato. Fortunatamente però le analisi condotte confrontando le tracce di input con quella in uscita hanno dimostrato i buoni risultati raggiunti in questo compito. Il modello sviluppato è certamente in grado di eseguire un NST tra due fonti audio mantenendo un'alta fedeltà con i riferimenti e imprimendo con una buona precisione lo stile di una traccia sull'altra con una quantità di rumore sostanzialmente trascurabile e dovuta principalmente alle tecniche di ricostruzione della fase.

4.2 Genre Classification

Dalle analisi condotte capiamo chiaramente come l'approccio al problema della classificazione dipenda strettamente dai dati in nostro possesso e dalle tecniche e gli strumenti utilizzati, producendo talvolta ottimi risultati e in altri casi risultati certamente migliorabili. Cerchiamo allora di trarre delle conclusioni in questo senso per ogni scelta intrapresa nell'affrontare questo problema.

4.2.1 Dati .csv

Il modello 1 potrebbe beneficiare di strategie di regolarizzazione e dell'introduzione di livelli di dropout per migliorare la sua performance sulla validazione. I modelli 3 e 4 potrebbero richiedere ulteriori ottimizzazioni per gestire l'overfitting. Il grafico del modello 4 ci mostra inoltre come la validation loss sia facilmente contenibile riducendo le epoche di allenamento. Infatti questa metrica è l'unica che tende a crescere col tempo mentre le altre si attestano velocemente al loro valore asintotico. Lo stesso discorso vale anche per il modello 1 dove la validation loss tende ad aumentare con le epoche anche se in maniera meno accentuata rispetto al modello 4.

4.2.2 Data Augmentation

Il processo di Data Augmentation è certamente complesso e richiede una conoscenza veramente approfondita dei dati. La chiara presenza di overfitting ci indica che i modelli sono certamente migliorabili, nonostante fare Data Augmentation sia già di per sè una tecnica atta a ridurre l'overfitting. Se consideriamo solo questo aspetto, l'indicazione principale sarebbe quella di andare a rivalutare le trasformazioni applicate alle immagini e alle tracce audio con l'intento di migliorare ancora di più la capacità di generalizzare dei modelli.

Image Augmentation

Possiamo ipotizzare che un modello del genere abbia quasi sicuramente bisogno di tecniche di regolarizzazione e di layer di dropout che in questo modello non sono stati inclusi. Potrebbe darsi anche che questi risultati siano dovuti alla complessità del modello, infatti una rete neurale con molti parametri potrebbe non essere comunque in grado di adattarsi ai dati in maniera efficiente. Ridurre le dimensioni del modello è certamente una soluzione da esplorare, in quanto il modello per l'Image Augmentation è quello con il maggior numero di parametri di allenamento tra tutti.

Audio Augmentation

Il discorso per l'Audio Augmentation invece è leggermente diverso. Le tecniche di dropout sono state applicate e potrebbero essere il motivo della maggiore precisione rispetto al modello di Image Augmentation, ma, vista comunque la capacità del modello di predire correttamente solo un genere su due, possiamo immaginarci che anche il dropout vada migliorato ulteriormente. Anche in questo caso l'aggiunta di tecniche di regolarizzazione come la L1 o la L2 potrebbero aumentare la precisione del modello.

4.3 Sviluppi futuri

Le implementazioni a cui ci siamo riferiti risalgono al periodo di Giugno 2022 e nell'arco di poco più di un anno l'intelligenza artificiale, soprattutto quella generativa, ha subito un'accelerazione incredibile dal punto di vista dell'innovazione. Dall'avvento dei modelli GPT (Generative Pretrained Transformer) l'intelligenza artificiale è diventata con tutta probabilità il trend maggiore dell'anno 2023 e mira ad esserlo anche per l'anno corrente. Viene da sé che anche nella sfera delle applicazioni musicali i miglioramenti sono stati notevoli. L'AI generativa allo stato attuale riesce a creare un brano musicale a partire da un testo e da una descrizione della base musicale desiderata, mentre i modelli e le tecniche di classificazione sono migliorati notevolmente.

Uno sviluppo plausibile per cercare di ammodernare il modello di NST potrebbe essere proprio quello di sfruttare il modello attuale per costruire un applicativo in grado di generare un brano, includendo anche la possibilità di inserire una parte cantata a partire da un testo. Una struttura del genere sarebbe realizzabile inserendo un layer di interpretazione del testo basato su un modello GPT che va a impartire alla rete convoluzionale le direttive per generare la base musicale. Al contempo un modello di text-to-speech[1] trasforma il testo in una fonte audio.

[1] Ad oggi i modelli più performanti in termini di text-to-speech si basano su reti RNN o LSTM

Infine un'ulteriore rete convoluzionale, applicando i princìpi visti in precedenza, va a unire la parte cantata con la base musicale con l'intento principale di far suonare a tempo la parte cantata. Per quest'ultima fase si potrebbe approfondire l'utilizzo di una rete LSTM (Long Short-Term Memory) in sostituzione di quella convoluzionale per via della migliore abilità nel codificare la struttura temporale e in modo da mettere a tempo la parte cantata. Ad oggi esistono già alcuni applicativi che svolgono questo compito e i modelli più utilizzati per raggiungere tale scopo sono Bark e Chirp [12]. Il primo è specializzato più sulla generazione della parte cantata mentre il secondo si concentra maggiormente sulla composizione musicale.

Il modello di NST richiede comunque un aggiornamento a livello di implementazione, soprattutto per via dell'introduzione di TensorFlow 2, avvenuta a seguito del completamento del progetto. Questo aggiornamento stravolge completamente le implementazioni di TensorFlow, rendendo utilizzabile il modello solamente in un ambiente aggiornato alla versione 1. Il processo richiederebbe quindi la migrazione[2] completa del codice per sfruttare le nuove interfacce di programmazione (API) di TensorFlow 2 e successivamente la valutazione del nuovo modello di NST.

A livello di performance l'obiettivo principale è sicuramente quello di mitigare ancora di più il rumore prodotto in uscita esplorando nuove soluzioni per la ricostruzione della fase, che abbiamo visto essere un importante collo di bottiglia in questo senso.

Nonostante invece il GC abbia mostrato ottime prestazioni per la classificazione tramite feature in formato tabulare, restano ancora diverse defezioni nella precisione per i modelli di Data Augmentation. Aumentare l'accuratezza di questi modelli andando ad agire sulle tecniche di trasformazione dei dati dovrebbe essere innanzitutto l'obiettivo primario, ma si potrebbero allo stesso tempo esplorare nuove architetture per quanto riguarda i modelli. Alcuni approcci plausibili potrebbero riguardare l'impiego delle Recurrent Neural Networks (RNN), ottime per gestire sequenze temporali come i flussi audio, oppure l'utilizzo di modelli versatili come la Random Forest o i Decision Trees.

Se si vuole ragionare invece su possibili sviluppi on-top, alcune soluzioni possono essere sicuramente quelle discusse brevemente nell'introduzione riguardanti i sistemi di raccomandazione di brani. L'idea di base potrebbe essere la costruzione di cluster di generi dove è possibile misurare le distanze tra una traccia e l'altra, in modo simile a ciò che fanno i modelli GPT con le parole. Infatti, i modelli di linguaggio GPT stimano la parola successiva (più precisamente il token successivo) in base alla sua distanza da quella precedente. In maniera analoga si potrebbe consigliare all'utente un brano in base alla distanza da un brano di riferimento.

In conclusione, le tecnologie trattate gettano sicuramente delle basi da cui partire per strutturare dei veri a propri applicativi, le cui funzionalità e i cui impieghi possono essere molteplici, dimostrando ulteriormente l'impatto che l'intelligenza artificiale può avere al giorno d'oggi.

[2] TensorFlow mette a disposizione degli script per facilitare il processo di migrazione: https://www.TensorFlow.org/guide/migrate

Bibliografia

[1] IMPF Ethical Guidelines on generative Artificial Intelligence, https://www.impforum.org/wp-content/uploads/2023/10/IMPF-Ethical-Guidelines-on-generative-AI-docx.pdf

[2] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, *"A Neural Algorithm of Artistic Style"*, https://arxiv.org/pdf/1508.06576.pdf

[3] Librosa STFT, https://librosa.org/doc/main/generated/librosa.stft.html

[4] Haochen Li, Yuyan Zhao, *"Neural Style Transfer Replication Project"*, https://www.cs.princeton.edu/courses/archive/fall17/cos429/COS429-proj/COS429_neuralstyle_HaochenYuyan.pdf

[5] Neural Transfer Using PyTorch, https://pytorch.org/tutorials/advanced/neural_style_tutorial.html

[6] Gram Matrix, https://en.wikipedia.org/wiki/Gram_matrix

[7] Griffin-Lim Algorithm, https://paperswithcode.com/method/griffin-lim-algorithm

[8] Frobenius Norm, https://en.wikipedia.org/wiki/Matrix_norm#Frobenius_norm

[9] Librosa Griffin-Lim, https://librosa.org/doc/main/generated/librosa.griffinlim.html

[10] George Tzanetakis, Georg Essl, Perry Cook, *"Automatic Musical Genre Classification Of Audio Signals"*, https://ismir2001.ismir.net/pdf/tzanetakis.pdf

[11] Yugesh Verma, *"A Tutorial on Spectral Feature Extraction for Audio Analytics"*, https://analyticsindiamag.com/a-tutorial-on-spectral-feature-extraction-for-audio-analytics/

[12] Suno AI: New AI Technology Can Generate Original Songs from Text, https://medium.com/illumination/suno-ai-new-ai-technology-can-generate-original-songs-from-text-df318c9e2c3e