Riconoscimento generi musicali

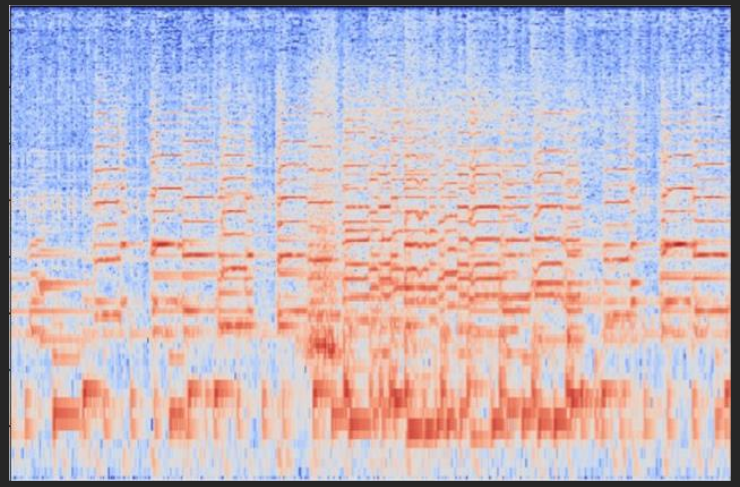
Bertoli Nicolò

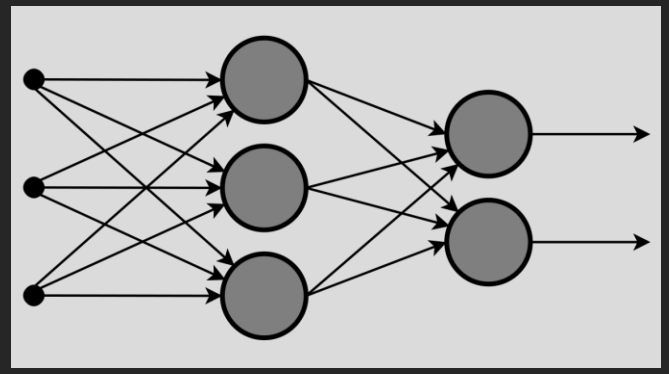
Obiettivi

Obiettivi

- Costruire un modello capace di riconoscere il genere musicale di un file audio.

Idea di base: utilizzare il riconoscimento di immagini applicato agli spettrogrammi





Jazz

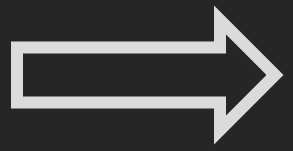
Blues

Metal

...

Obiettivi

- Costruire un' applicazione mobile che permetta di utilizzare il modello in modo semplice.



File audio da 15 secondi

Stato dell' arte

Stato dell' arte

Esistono varie ricerche in questo ambito, ma...

- Non esistono app ben funzionanti per la classificazione dei generi musicali.

- Le ricerche che ho individuato utilizzano tutte audio più lunghi (30s)

Stato dell' arte

Esistono varie ricerche in questo ambito, ma...

- Nella maggior parte delle ricerche vengono utilizzati pochi generi musicali molto generici, mentre io ne volevo utilizzare di più specifici.

Per esempio spesso viene utilizzata la categoria "elettronica", che in racchiude sottogeneri molto diversi tra loro.

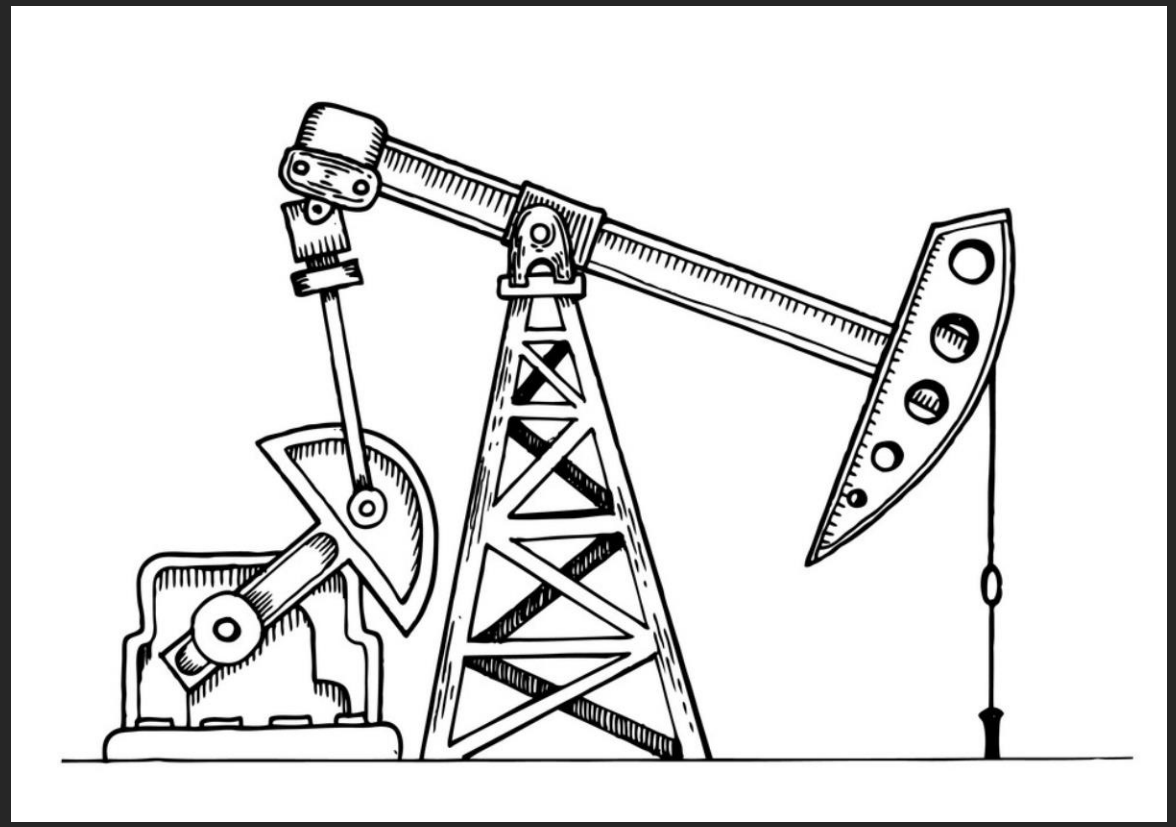
Strumenti utilizzati

Wavepad

Librosa

Tensorflow e Keras

Estrazione e raffinamento dati



Costruzione del dataset

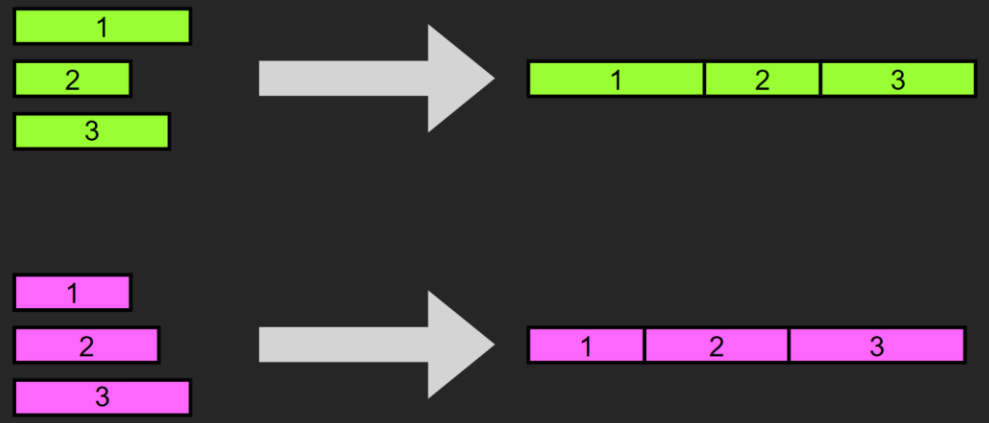
Generi utilizzati

Blues	**Classica**	**Dubstep**
House	**Jazz**	**Metal**
Pop-punk	**Rap**	**Reggae**

Costruzione del dataset

Unione album

Album divisi in singoli file audio (mp3 a 128 kbps)

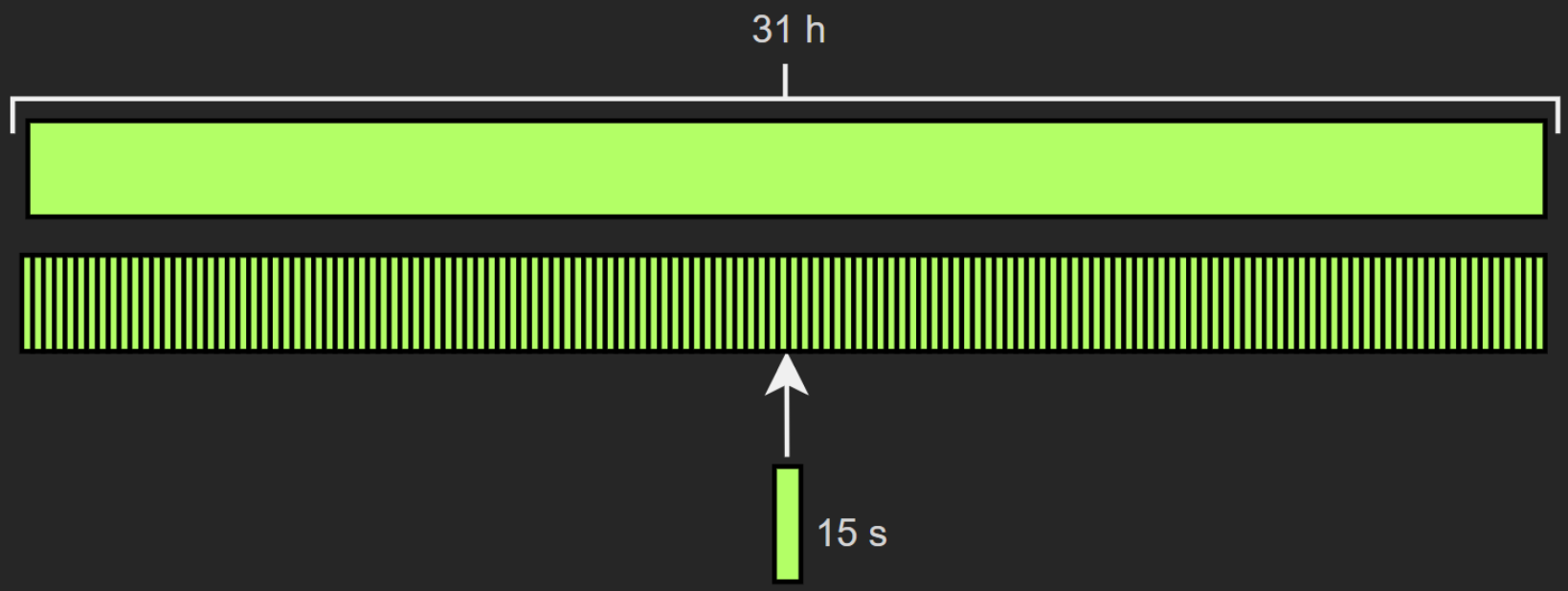


Un unico file audio da 31h per genere

(Circa 280 ore in totale)

Costruzione del dataset

Divisione della traccia di ogni genere in file audio da 15 secondi



Costruzione dataset

Chunking

Problema: la dimensione dei dati necessaria ad addestrare la rete neurale supera la capacità della mia RAM.

Quindi non posso caricare in memoria tutto il mio training set e poi chiamare la fit().

La soluzione è utilizzare il chunking: divido il mio train set in chunks (insiemi di elementi) e poi chiamo la fit passandole un chunk per ogni genere musicale.

Ho usato 60 chunks per il train set, 11 per il test set e 3 per il validation set.

Costruzione dataset

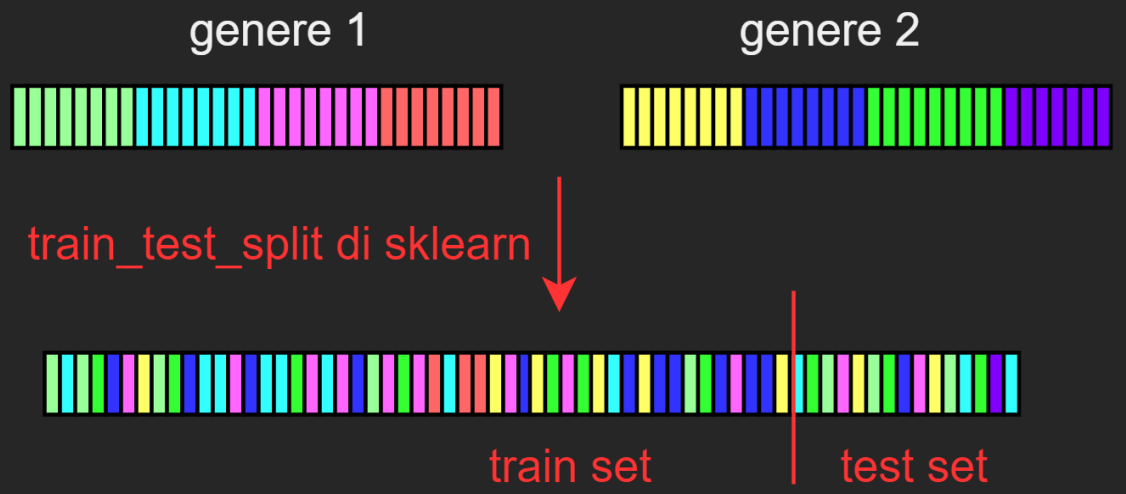
chunking

Problema: gli spettrogrammi non sono tutti indipendenti tra loro, ma alcuni, quelli vicini appartengono a alle stesse canzoni/agli stessi album.

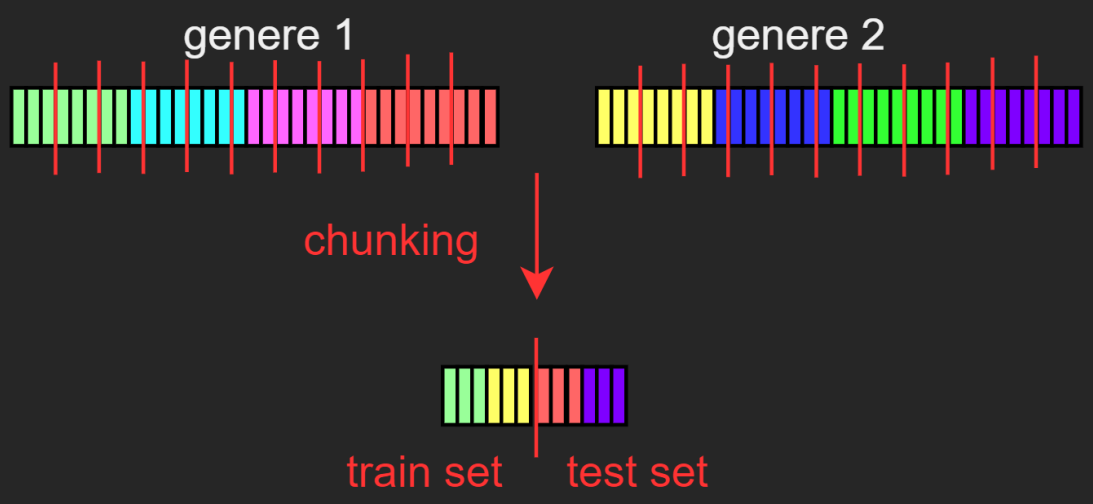
Di conseguenza non posso affidare ad un algoritmo (es train_test_split di sklearn) la divisione del mio dataset in test e train set, infatti in questo modo andremmo ad avere pezzi delle stesse canzoni nel train e nel test set, cosa che comprometterebbe i risultati.

La divisione del dataset in chunks risolve anche questo problema

Tracce degli stessi album in train e test set



Train e test set hanno tracce di album differenti

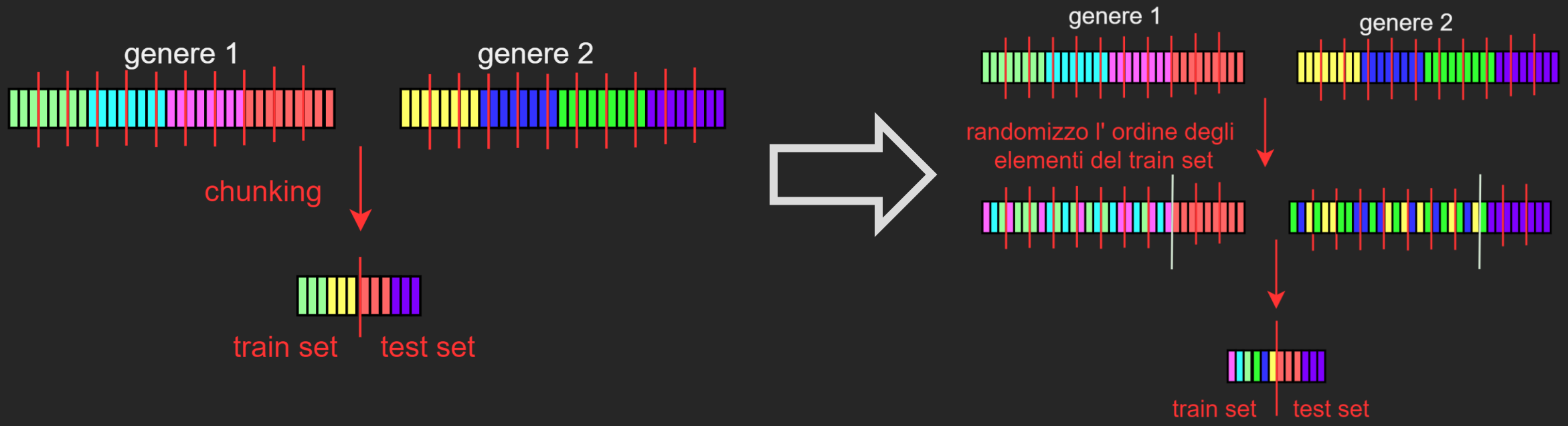


Costruzione dataset

chunking

Problema: dato che la rete si allena su chunks, e dato che ogni chunk corrisponde circa ad un album, in questo modo rischiamo che la rete ad ogni fit si vada ad adattare troppo all' album del chunk.

Soluzione: creo uno script per randomizzare l' ordine degli spettrogrammi del test set.

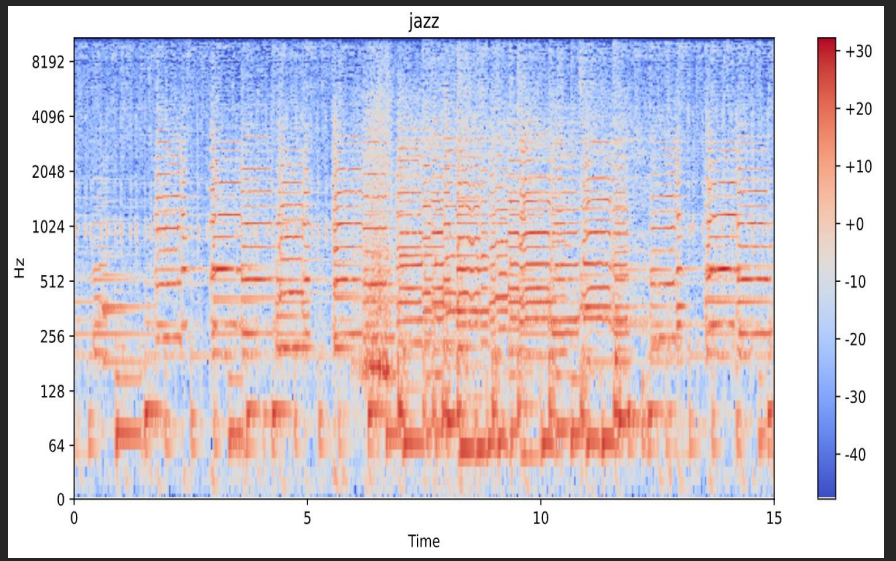


Costruzione dataset

Estrazione spettrogrammi

tracce audio
(formato .wav)

librosa



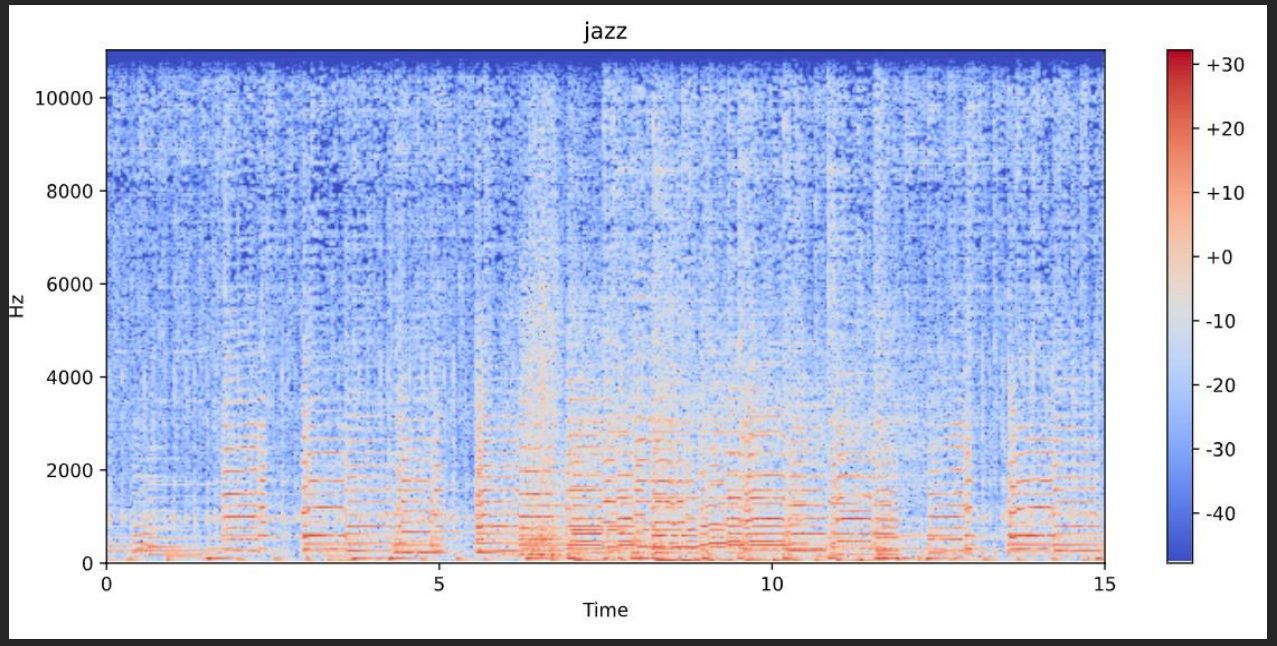
Spettrogramma
(numpy array)

Costruzione dataset

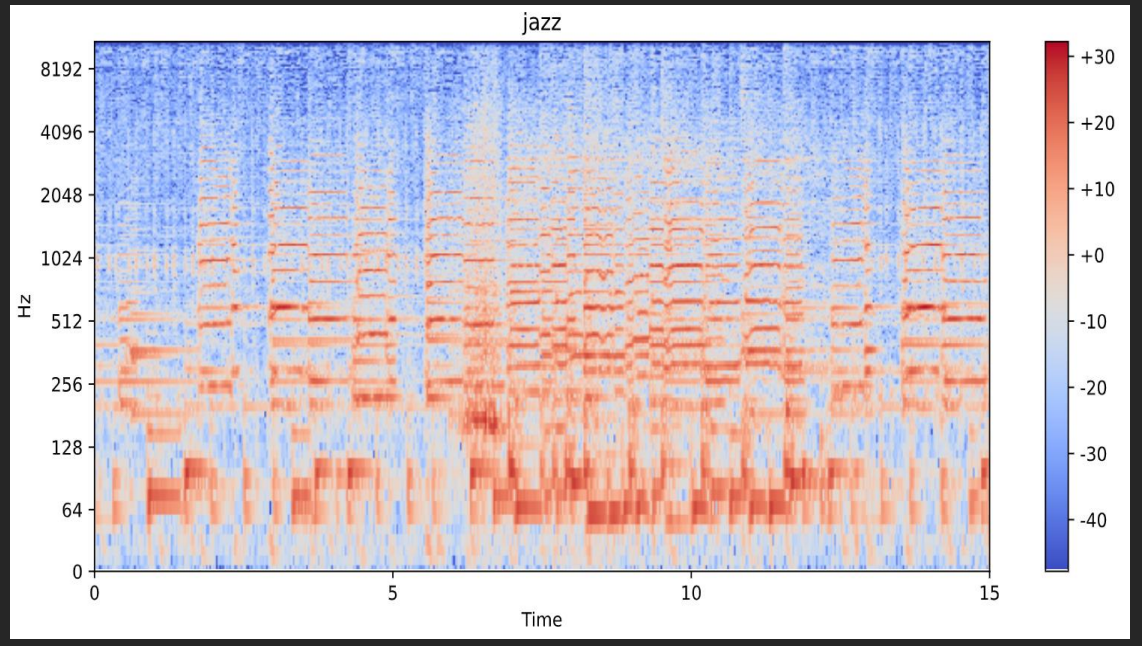
Scelta spettrogramma

Possibilità per il tipo di spettrogramma:

lineare

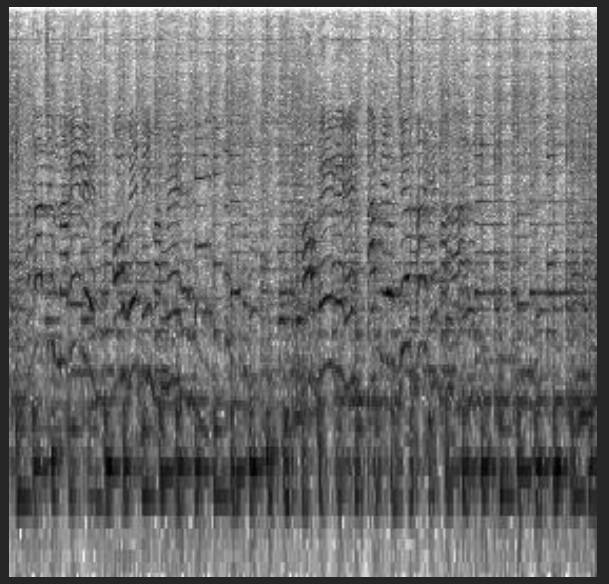


logaritmico



Costruzione dataset

Scelta spettrogramma

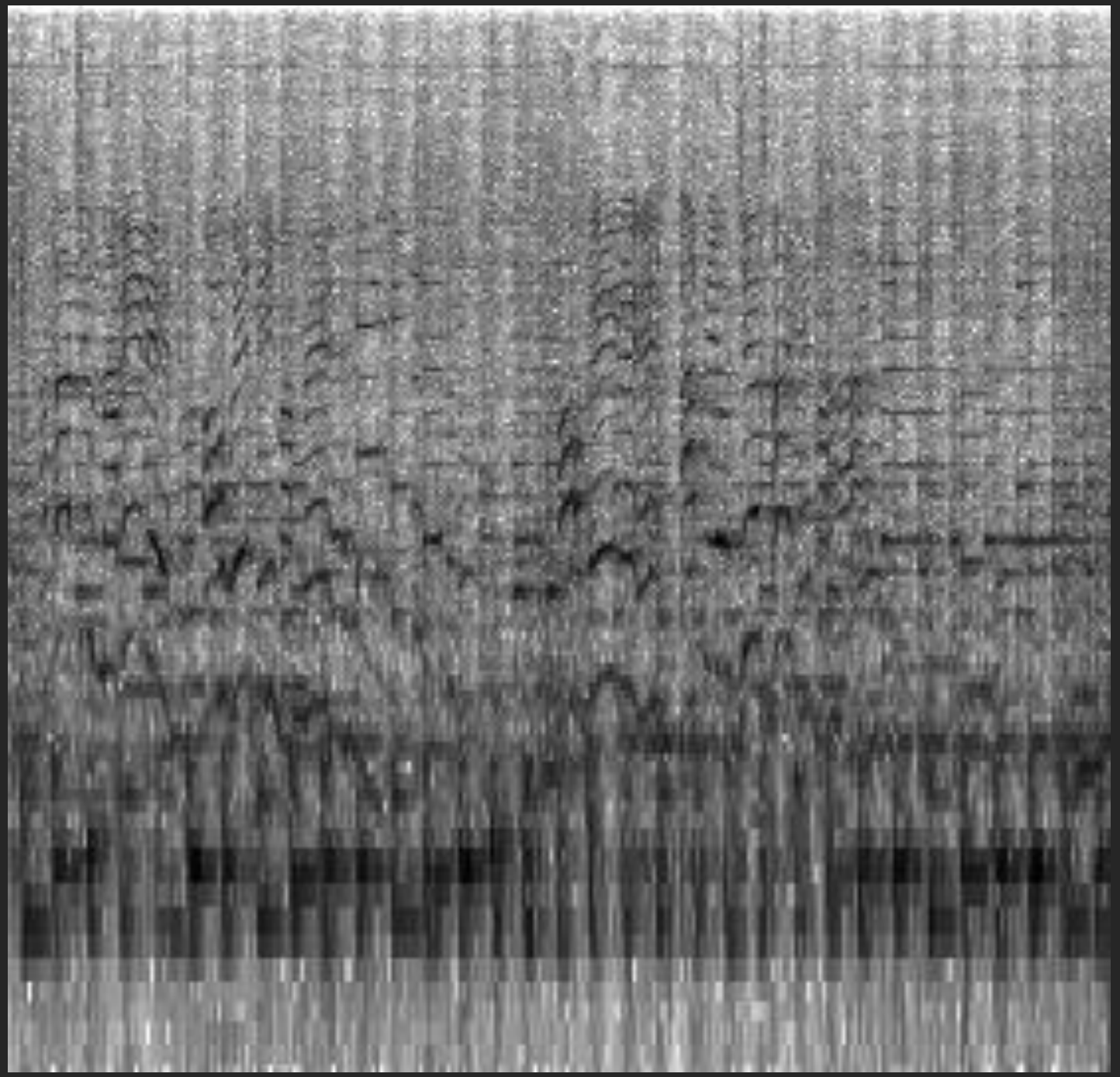


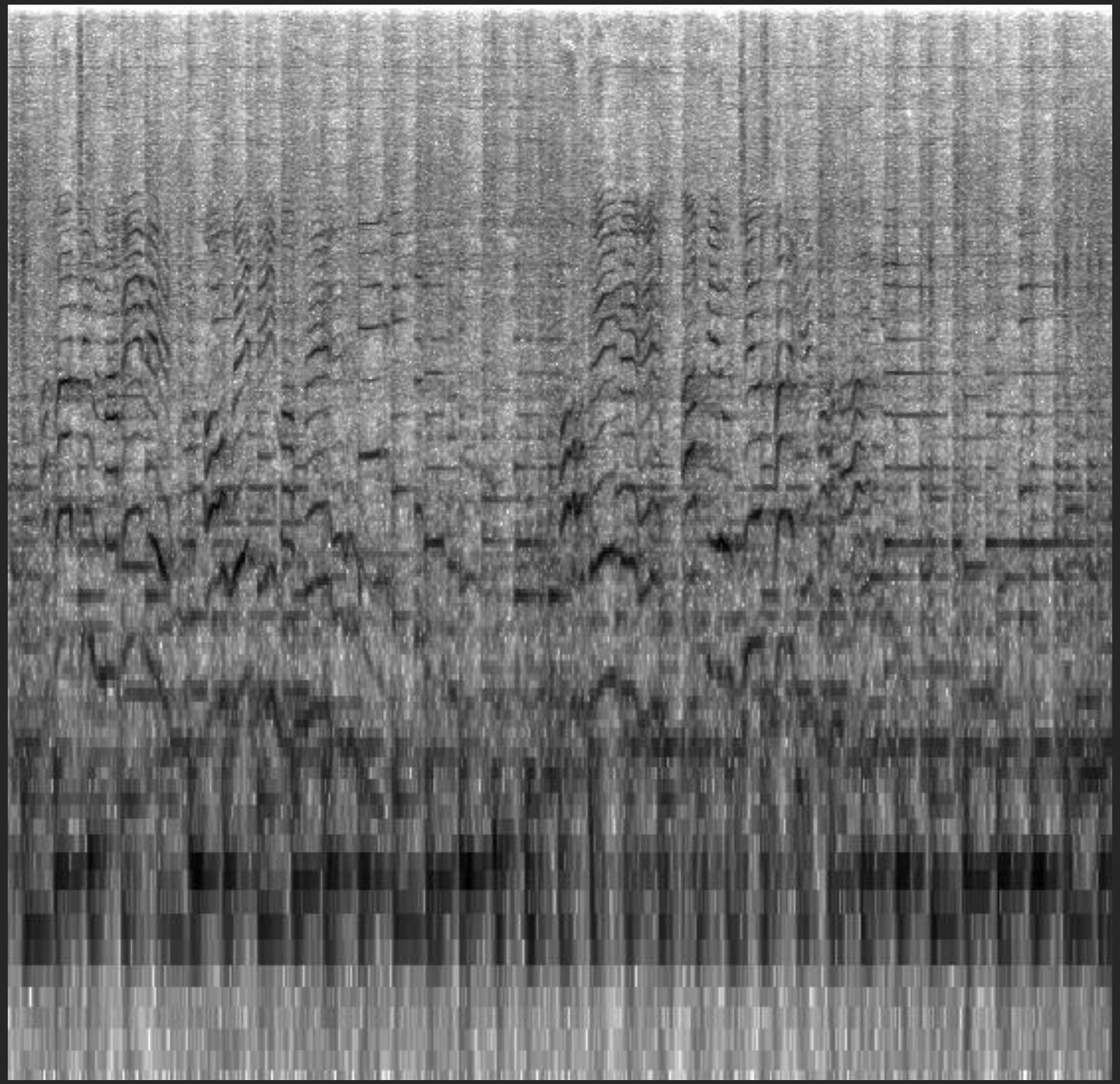
Quale risoluzione utilizzare ?

Possibili risoluzioni individuate per gli spettrogrammi:

- 279 x 271
- 558 x 543

Scelgo la risoluzione più bassa perché permette di avere buoni risultati mantenendo i tempi di addestramento accettabili (≃3h).

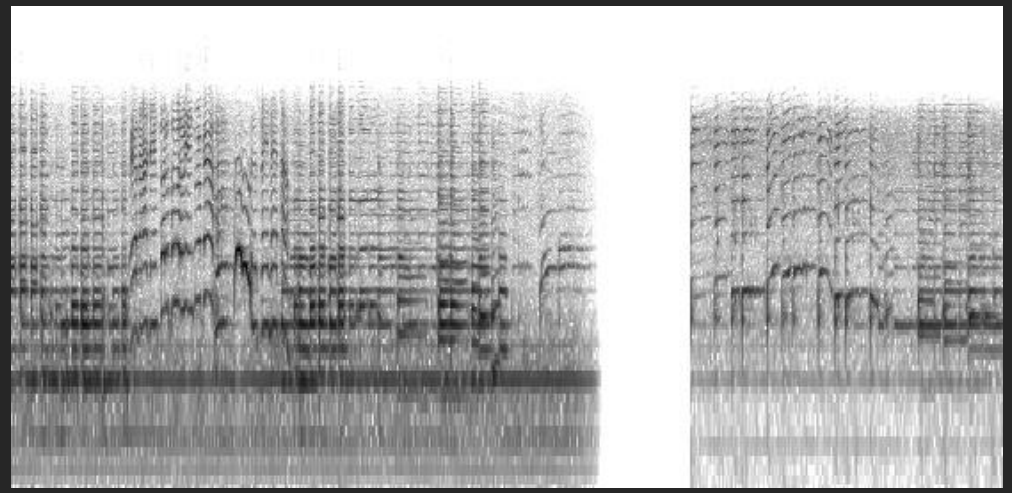




Costruzione dataset

Rimozione silenzi

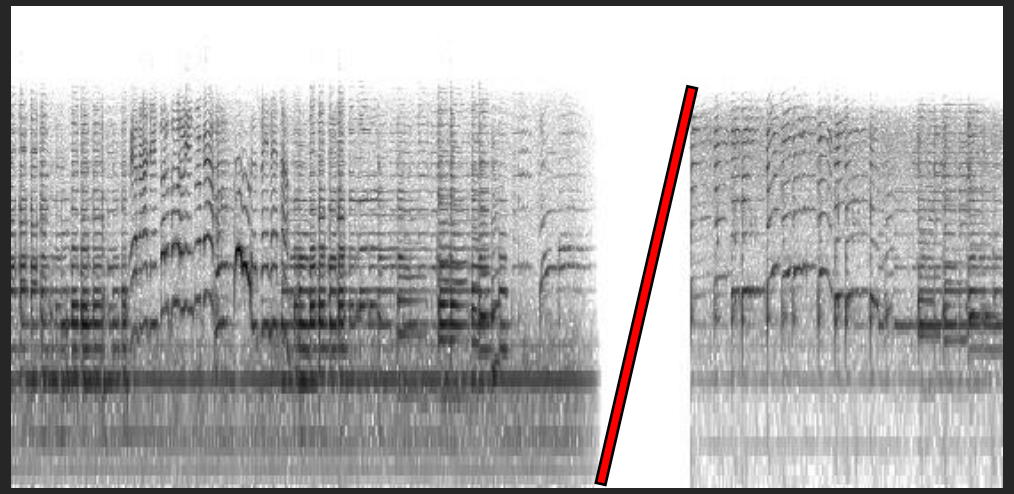
Se saltassimo questa fase, alcuni degli spettrogrammi che andremo a produrre ci apparirebbero cosi:



Questi spazi bianchi sono dovuti al silenzio tra una traccia e l' altra e costituiscono rumore che va a confondere la rete neurale.

Costruzione dataset

Rimozione silenzi



Problema: i silenzi potrebbero essere una feature utile a riconoscere alcuni generi

Esempio -> nella musica classica ci possono essere momenti di silenzio interni alle canzoni.

La soluzione ottima sarebbe quella di rimuovere solo i silenzi tra le canzoni, senza toccare quelli interni.

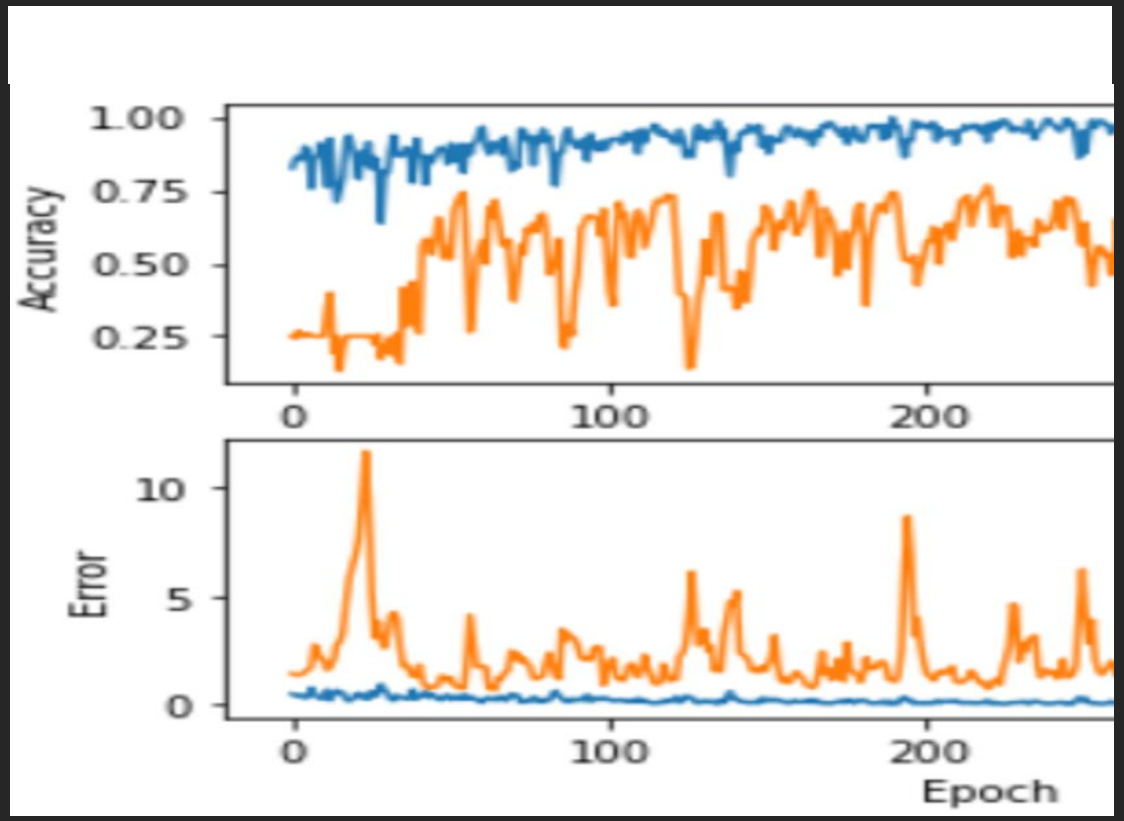
per cercare di limitare questa cosa, rimuovo solamente i silenzi superiori ad un secondo.

Costruzione dataset

Rimozione silenzi

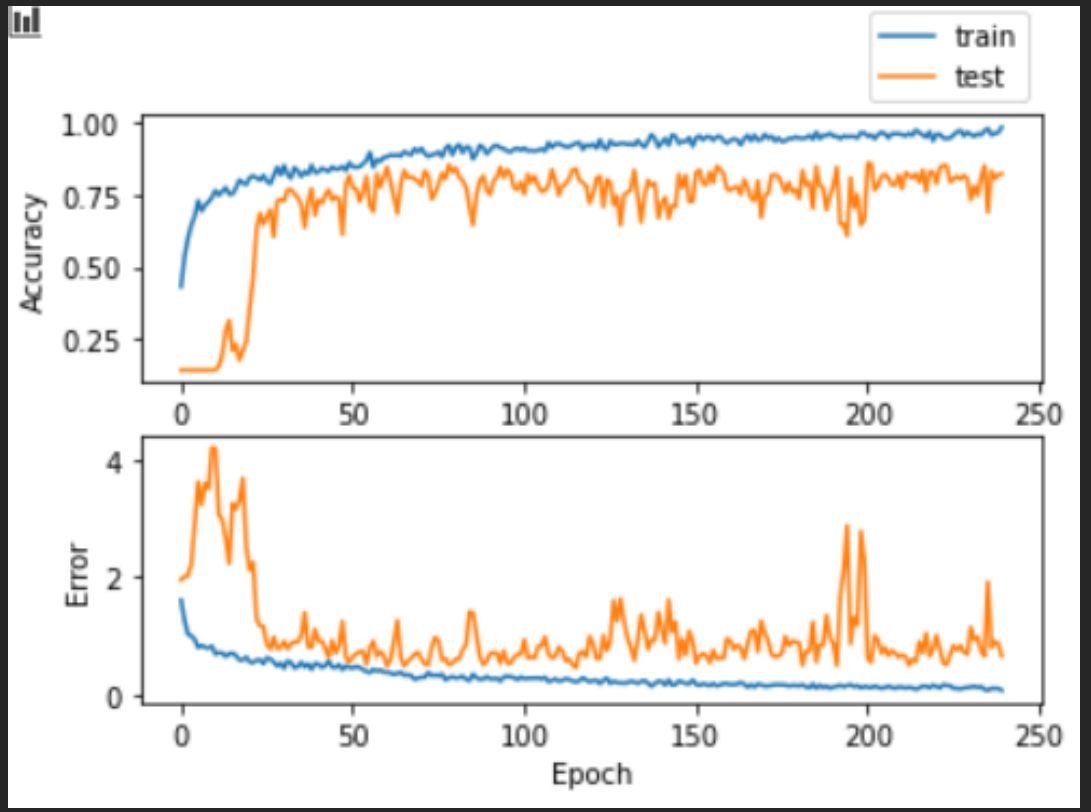
Rimuovere i silenzi porta ottimi risultati:

4 generi musicali



Accuracy test set ≃ 77.5%

7 generi musicali



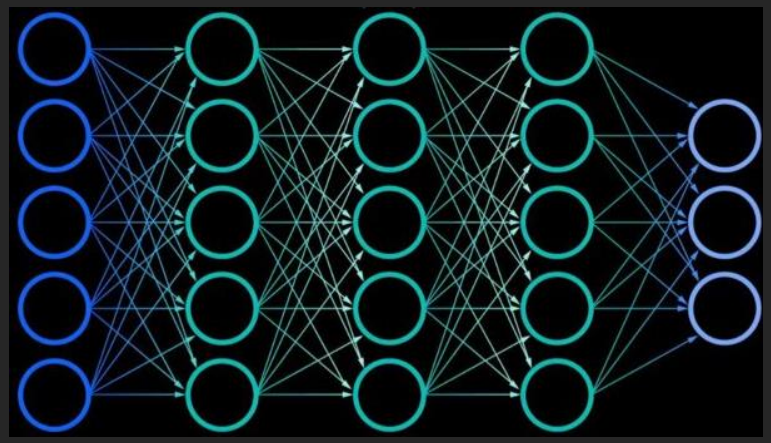
Accuracy test set ≃ 80.5%

Costruzione del modello

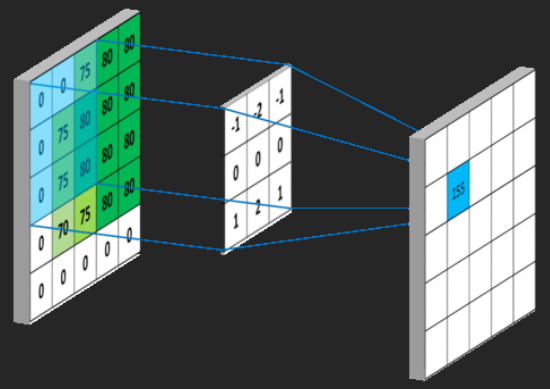
Costruzione modello

Scelta del modello

Rete neurale classica



CNN

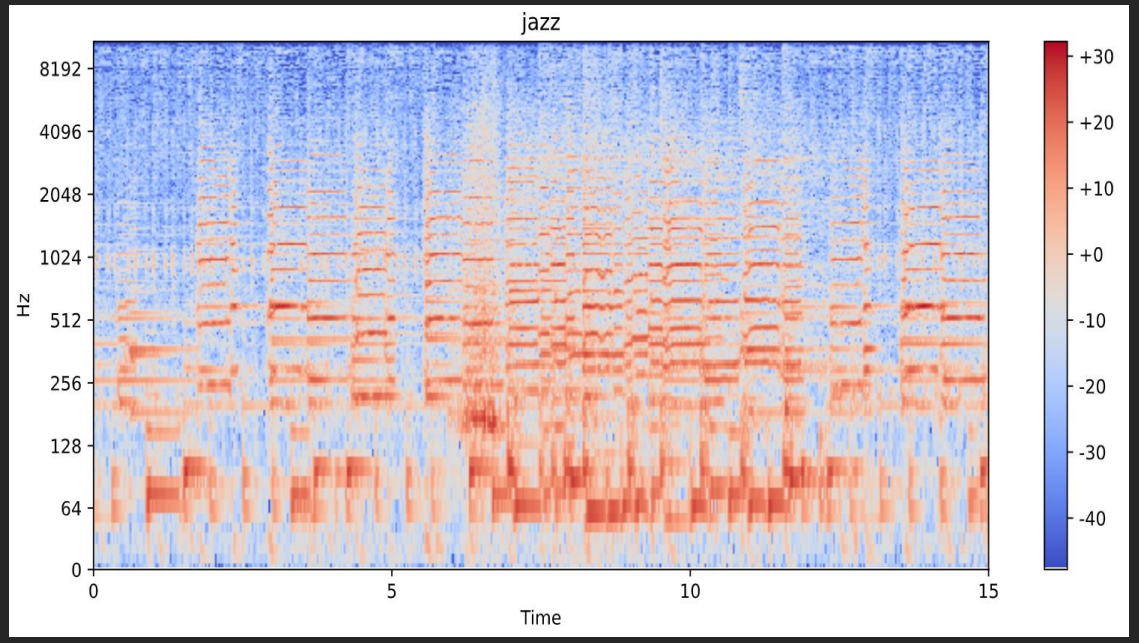


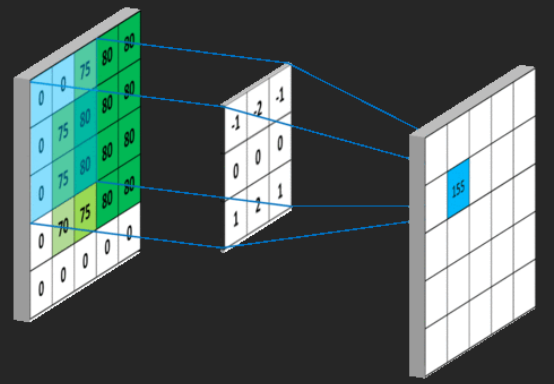
Costruzione modello

Scelta del modello

Decido di utilizzare la CNN con gli spettrogrammi logaritmici perché:

- Risultati leggermente migliori
- Tempi di addestramento più bassi.





Costruzione modello

La mia CNN

conv2d (32 kernel 3x3, strides = 1)
max_pooling2d (6x6, strides = 6)
batch_normalization

---

conv2d (32 kernel 3x3, strides = 1)
max_pooling2d (3x3, strides = 3)
batch_normalization

---

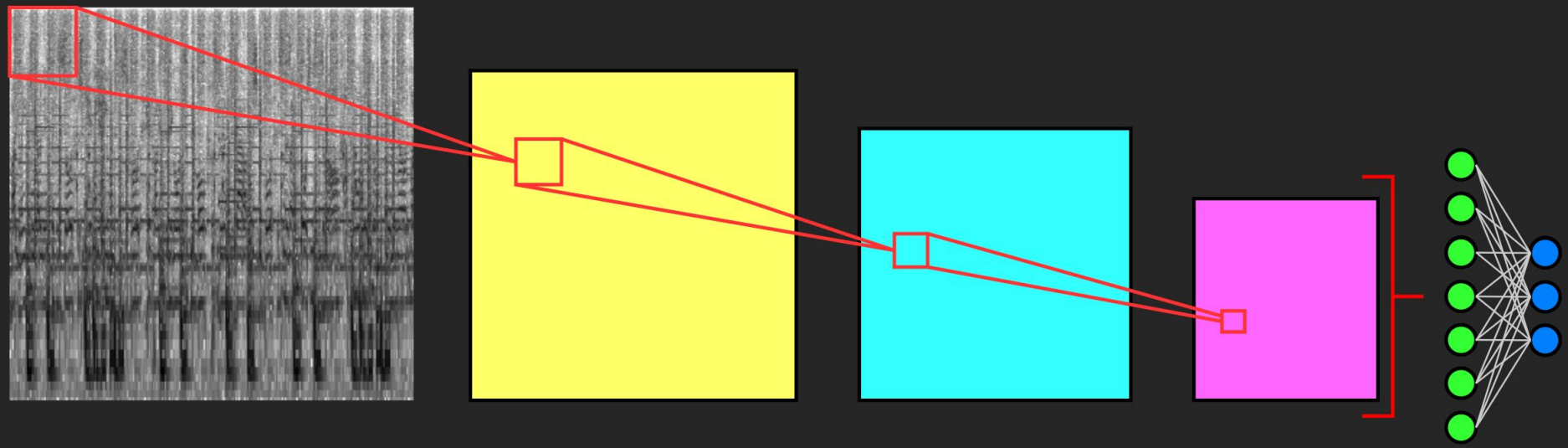
conv2d (32 kernel 2x2, strides = 2)
max_pooling2d (2x2, strides = 2)
batch_normalization

---

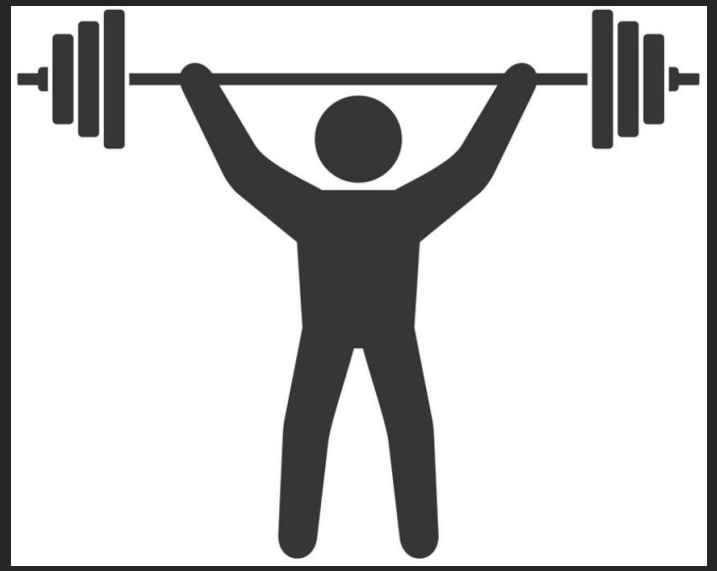
flatten
dense (64 nodi)
dropout (0.5)

---

dense(9 nodi)



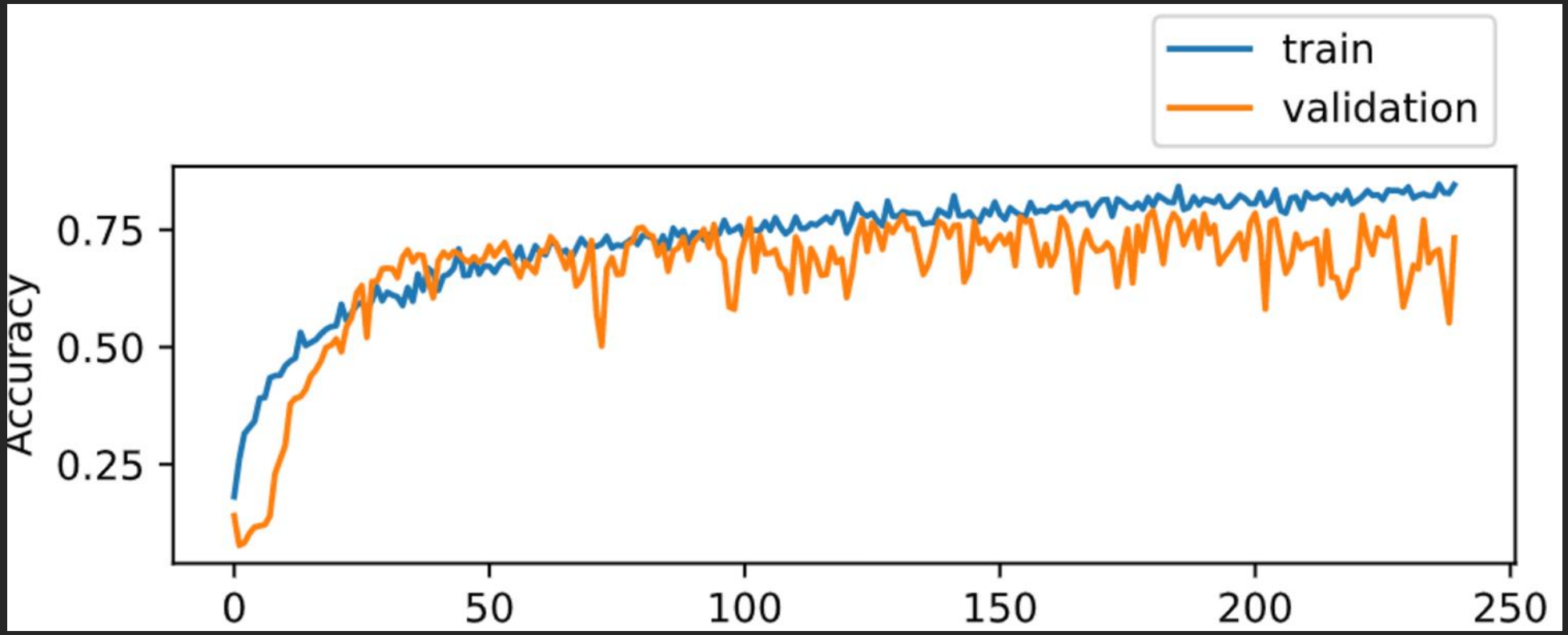
Addestramento modello



Addestramento modello

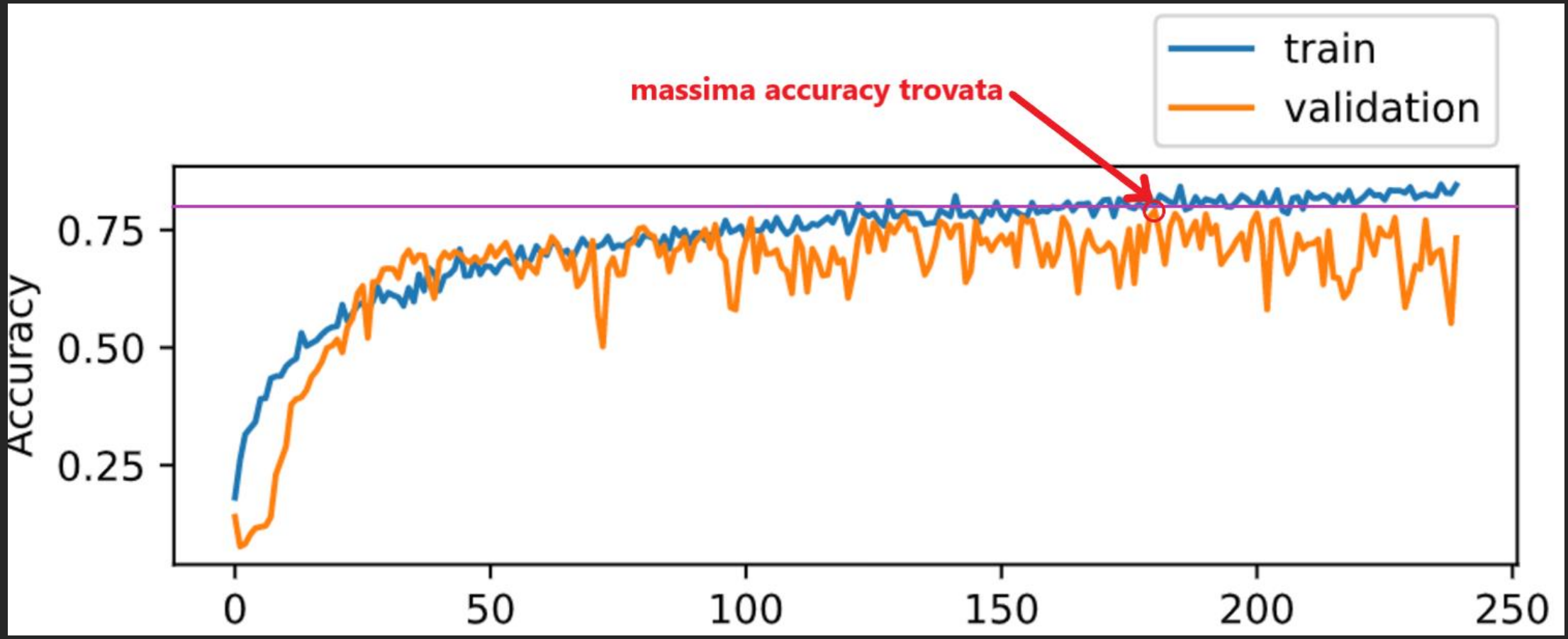
I primi test mostrano una grande variabilità dell' accuratezza nel tempo

Addestrare la rete più volte nelle stesse condizioni può portare a risultati molto diversi, a seconda del momento in cui si ferma l' addestramento.



Costruzione modello

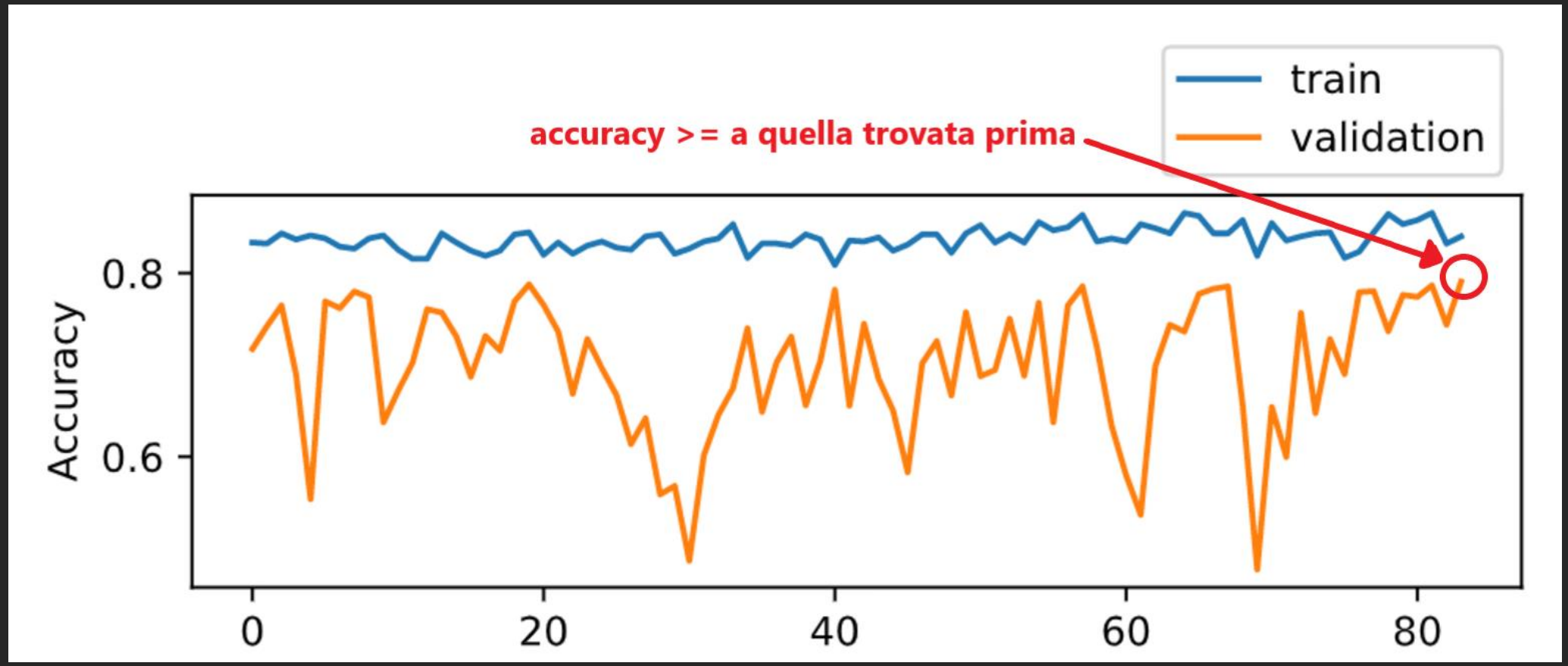
Fase 1: ricerca dell' accuracy massima possibile in un certo numero di epochs



Costruzione modello

Fase 2: continuo l' addestramento fin quando non ritrovo l' accuracy ottenuta in fase 1

Tengo il modello ottenuto in questo momento



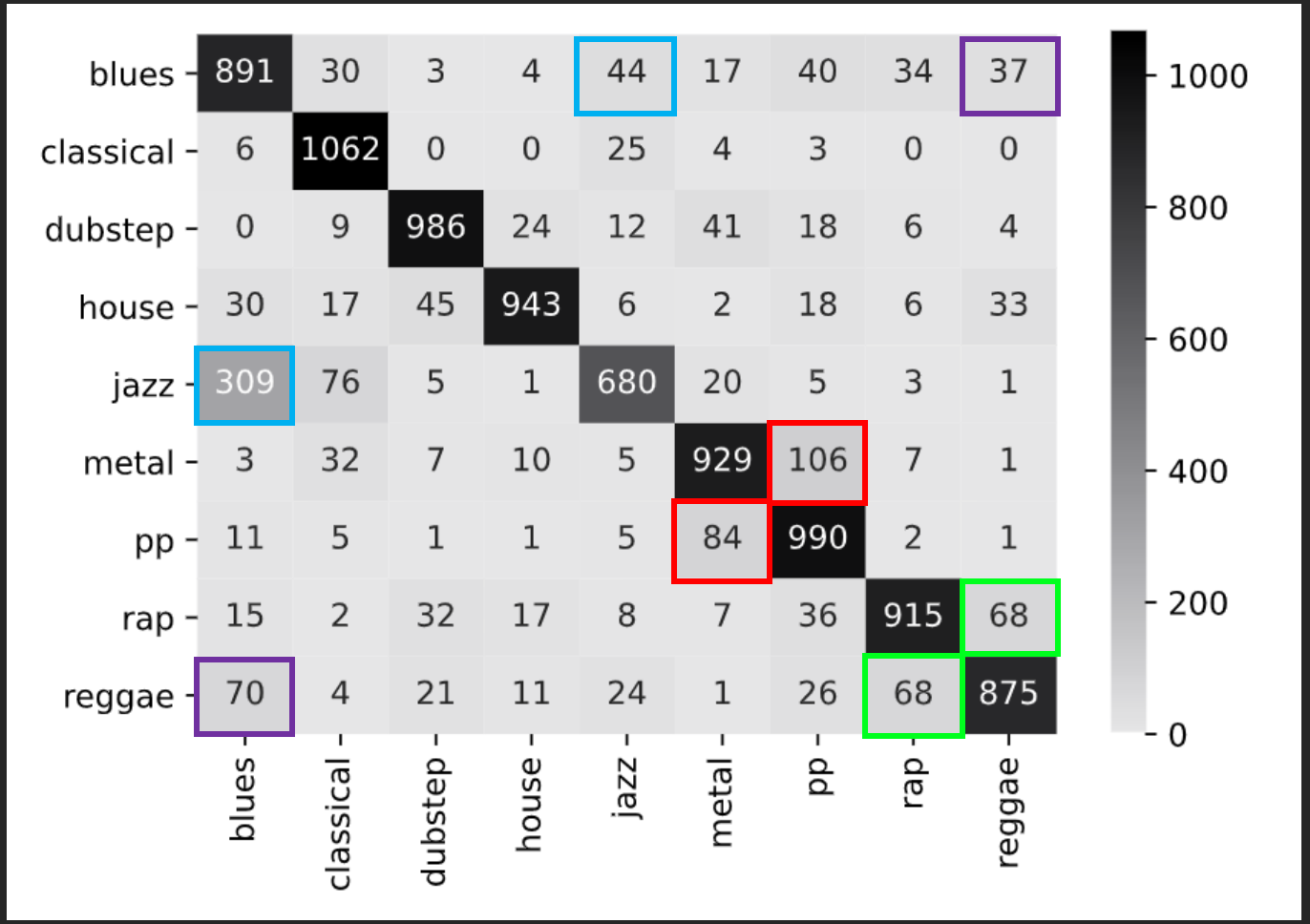
Risultati

Risultati

Accuracy validation set: 80%

Accuracy test set : 83.5%

Accuracy train set: 88.8%



Risultati

Test su alcuni sottogeneri del metal

Accuracy validation set: 81%

Accuracy test set : 79%

Accuracy train set: 92%