Music Genre Recognition using Neural Networks

Machine Learning Project

Professor: Andrea Asperti
Student: Andrea Poltronieri

Music Information Retrieval

Music information retrieval (MIR) is the interdisciplinary science of retrieving information from music.

MIR is a small but growing field of research with many real-world applications.

In recent years, deep learning methods have become more popular in the eld of music information retrieval (MIR) research. For example, while there were only 2 deep learning articles in 2010 in *ISMIR* conferences 1 and 6 articles in 2015 , it increases to 16 articles in 2016.

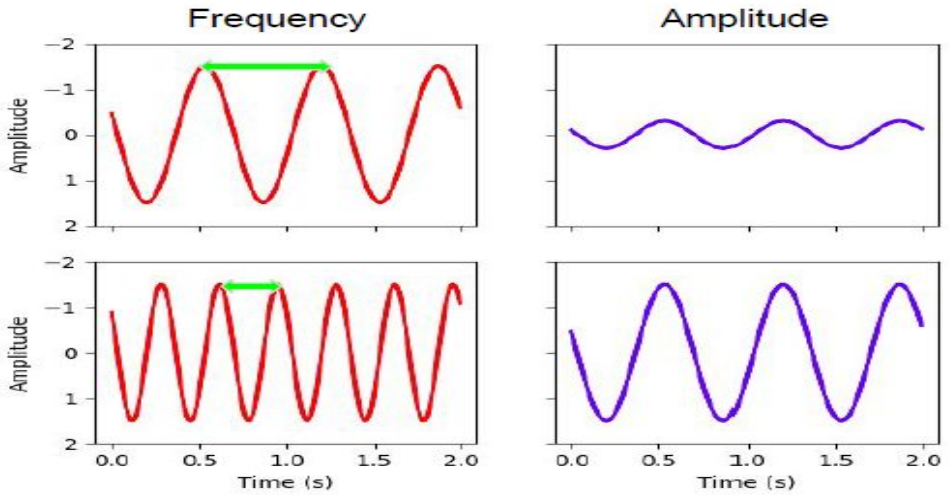
Audio Data Representation

One of the biggest issues in MIR concerns with how to represent music data.

I provide some examples in order to explain the path followed for making this project.

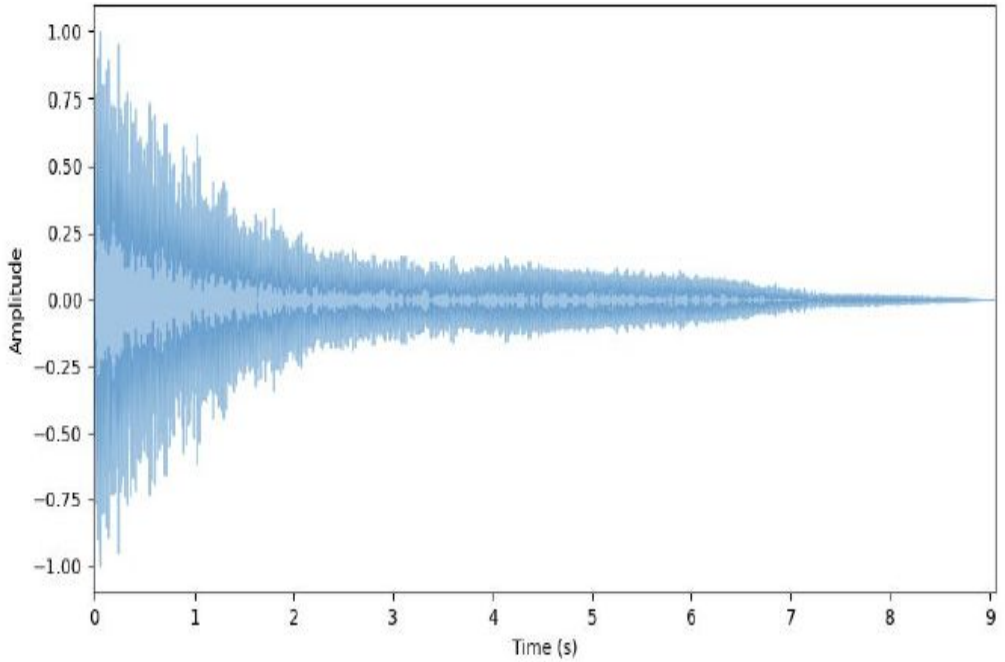
Sound Features

- **Frequency →** $f = \frac{1}{T}$
- **Amplitude →** related to loudness



Soundwave (row audio)

- x axis ➜ **Time** (s)
- y axis ➜ **Amplitude**



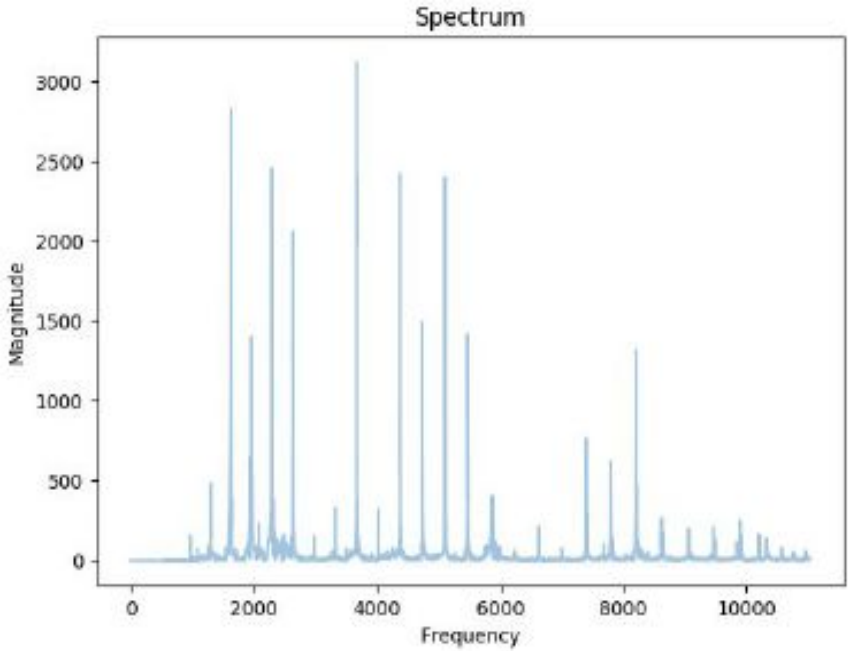
Fourier Transform

From time domain to frequency domain

Decompose complex periodic sound into sum of sine waves oscillating at different frequencies

- x axis → **Frequency**
- y axis → **Magnitude**

No time information



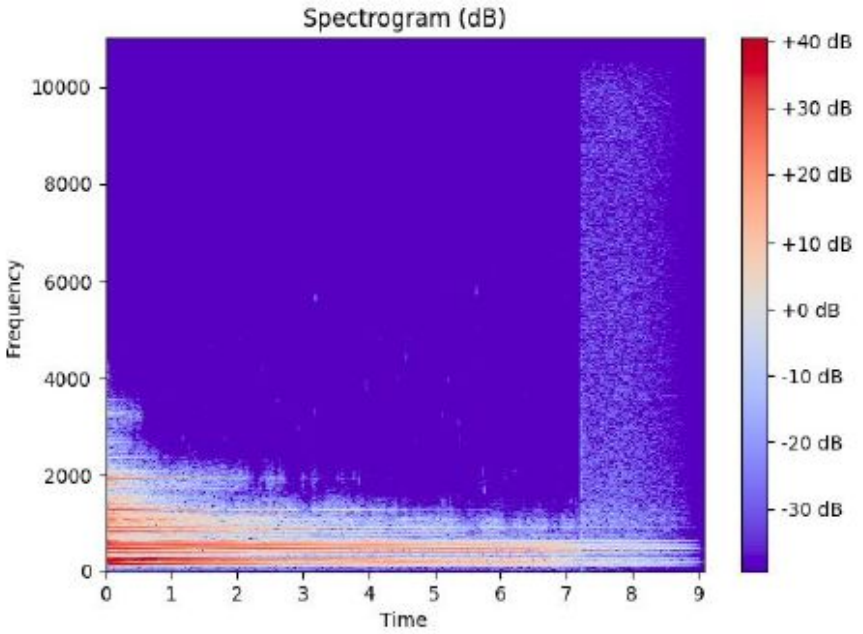
Short Time Fourier Transform (STFT)

Computes several FFT at different interval

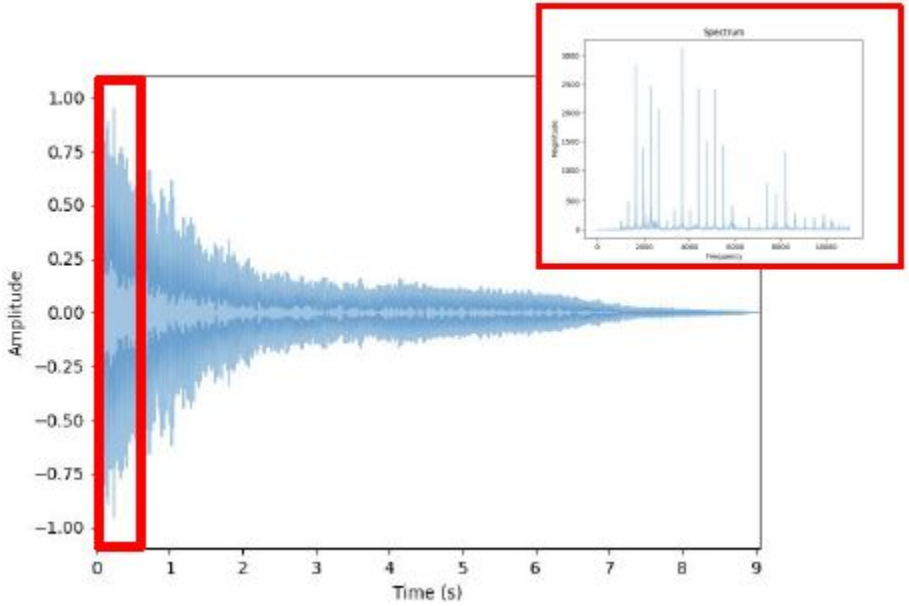
Preserves time information

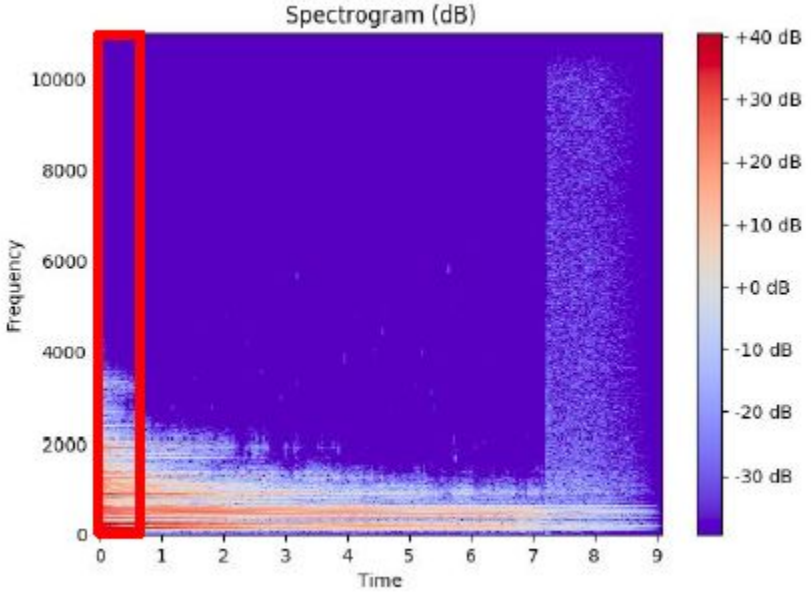
Fixed frame size (e.g., 2048 samples)

Gives a spectrogram (**time** + **frequency** + **magnitude**)



Short Time Fourier Transform (STFT)





Mel Frequency Cepstral Coefficients (MFCCs)

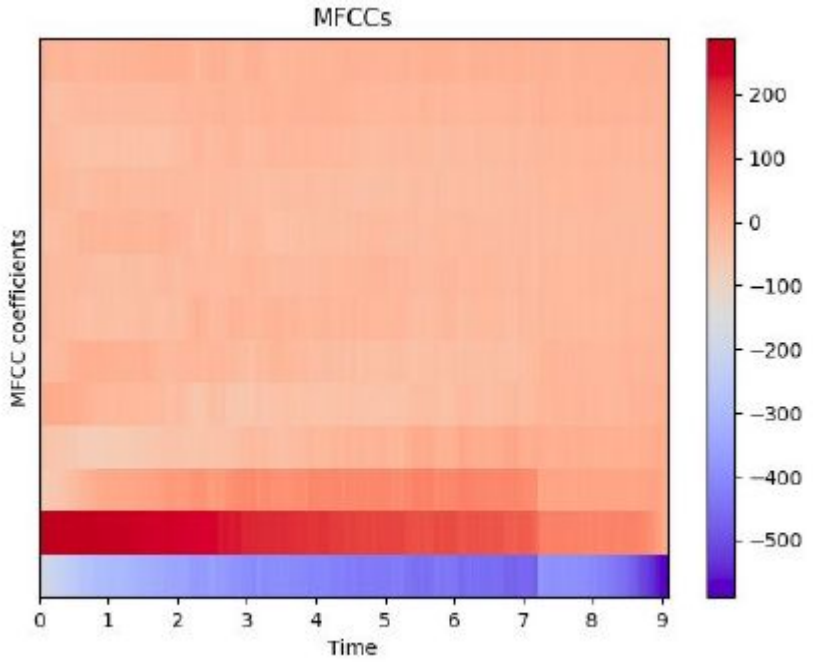
Capture timbral/textural aspects of sound

Frequency domain feature

Approximate human auditory system

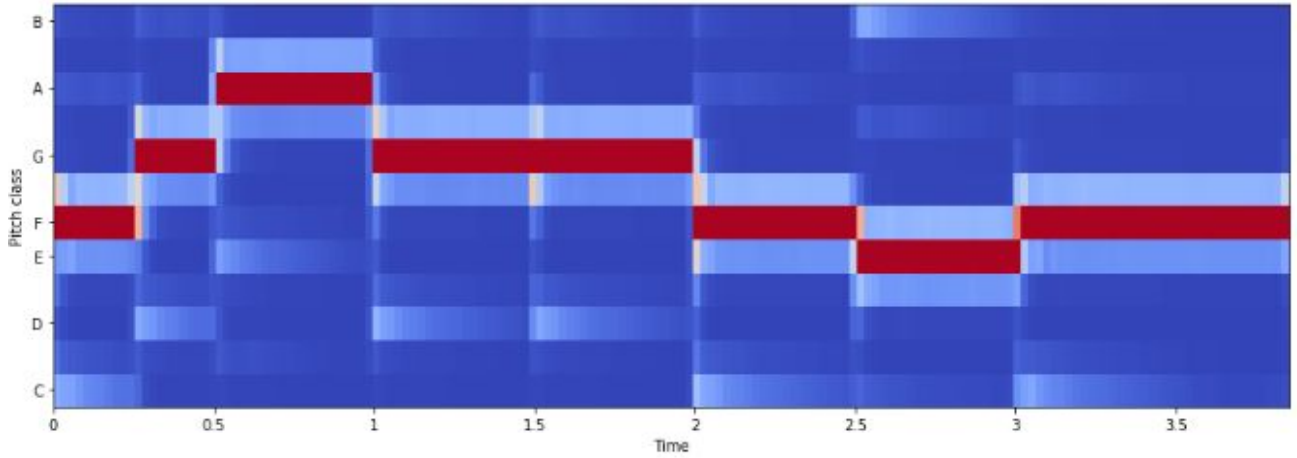
13 to 40 coefficients

Calculated at each frame



Chroma Features

Provides the energy distribution on a set of pitch classes, often with the western music's 12 pitches.



Tools and Libraries

Librosa was the library I used to import and extract all the feature mentioned in the previous slides.

The parameters Librosa takes as input for extracting MFCCs are:

```
mfcc(signal, sample_rate,num_mfcc, num_fft, hop_length)
```

All the features have been extracted and saved as a .json file.

For the training I have used **Google Colaboratory** with GPU acceleration, due to the big amount of data (even with such small datasets).

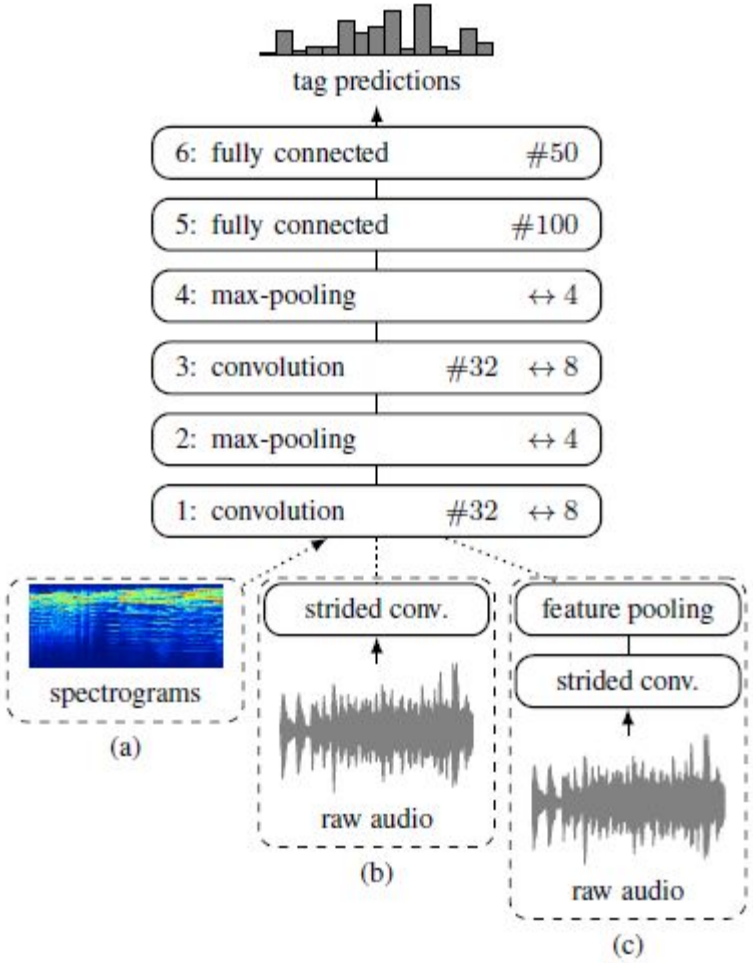
All the experiments were made using **Keras library**

The dataset was split into train, test (25%), and validation (20%) sets by using train_test_split from sklearn library.

Project’s starting paper

S. Dieleman and B. Schrauwen, “**End-to-end learning for music audio**,” in Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference

- uses both row audio and spectrograms
- on Magnatagatune dataset



Starting architecture

- 6 sequential **layers**:
  - 1 convolutional layer + 1 max pooling ➜ activation = “relu”
  - 1 convolutional layer + 1 max pooling ➜ activation = “relu”
  - 1 fully connected layer ➜ activation = “relu”
  - 1 fully connected layer ➜ activation = “softmax”
- optimizer ➜ adam
- input shape ➜ np.array with shape (n_segments, num_mfcc_vector_per_segment, mfcc_components, channel [=1])

1 - Changing Dataset

The original dataset (**Magnatagatune**) used for the original project was quite big:

- 25863 29-second audio clips;
- 16 kHz sample rate;
- annotated with 188 tags, only the main 50 have been considered.

I tried to use a much smaller dataset - **Marsaya**’s **GTZAN**:

- 1000 30-seconds clips;
- 22 kHz sample rate;
- annotated with 10 tags: ["blues", "classical", ”country", "disco", "hiphop", "jazz", "metal", "pop", "reggae", "rock"]

1 - Changing Dataset

To overcome the lack of data of the GTZAN dataset I **split every track in 10 segments**.

Then, I calculated the MFCCS for each segment, which have been associated to a separate label (as well as it was a separate track).

I decided to only consider the **MEL spectrogram (MFCC)**, since as demonstrated in the paper is performing better.

As proposed in the paper I made several trials with **different parameters** for the MEL spectrogram. Moreover I played with the number of **MFCCs components**, which in the paper was fixed to 128.

1 - Changing dataset - Results (1)

These are the results relative to the different parameter adopted, using the original net architecture proposed in the paper:

n_stft	hop_lenght	mel_components	val_accuracy
2048	1024	128	0.56
2048	1024	40	**0.64**
2048	1024	13	ERROR
1024	1024	128	0.38
1024	1024	40	0.49
2048	512	40	0.31
2048	512	13	ERROR
1024	512	40	0.30
512	512	40	0.52
512	256	40	0.29

1 - Changing dataset & Architecture

Experiment using 3 Conv2D layers and 1 dense layer:

n_stft	hop_lenght	mel_components	val_accuracy
2048	1024	128	0.75
2048	1024	40	0.75
2048	1024	13	**0.78**
1024	1024	128	0.76
1024	1024	40	0.75
2048	512	40	0.73
2048	512	13	0.74
1024	512	40	0.74
512	512	40	0.69
512	256	40	0.66

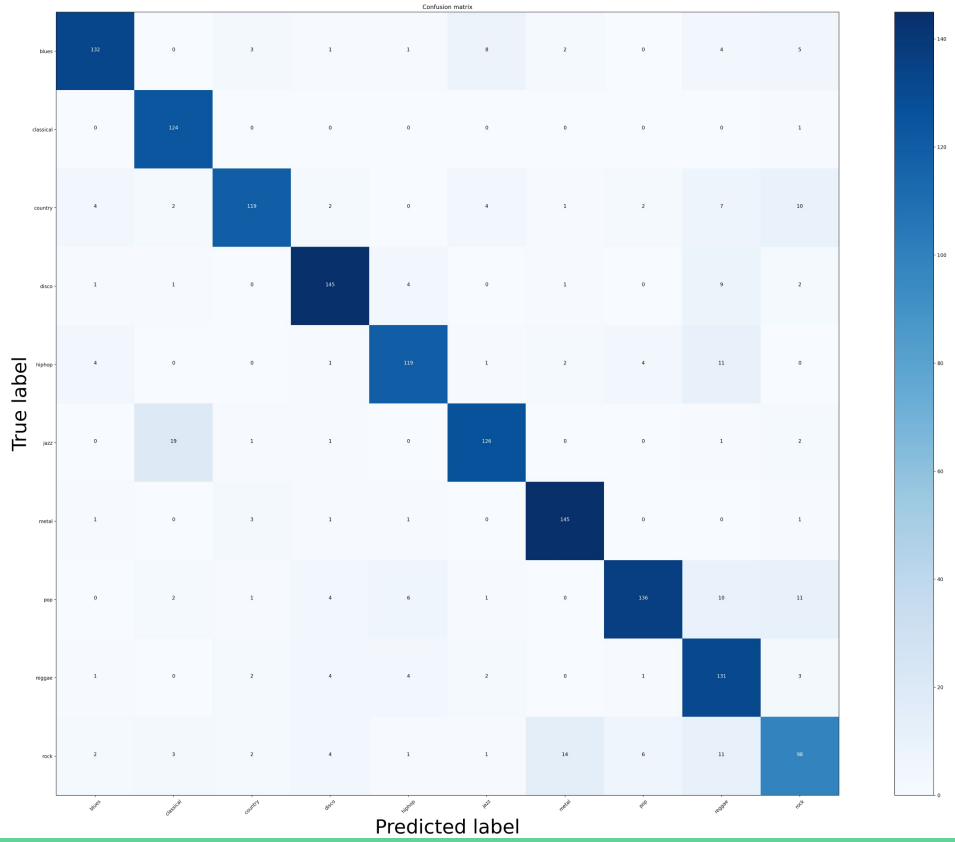
1 - Changing dataset & Architecture

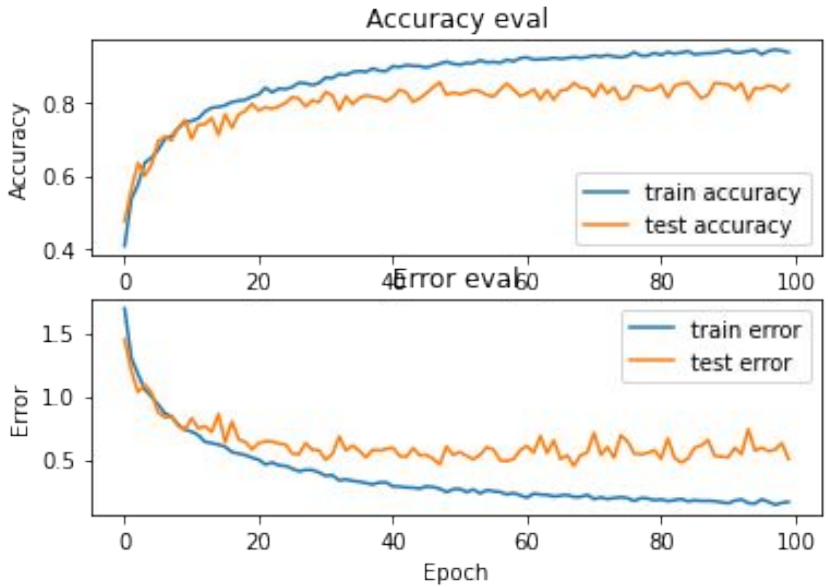
I made many experiments modifying the net architecture comparing the results of the different parameters showed in the last slides:

CNN_1	CNN_2	CNN_3	CNN_4	DENSE_1	DENSE_2	DENSE_3	**BEST RESULT**
128(4, 4) + MP (4, 4)	64(3, 3) + MP (3, 3)	32 (2, 2) + MP (2, 2)	-	128 w/ 0.3 dropout	10	-	0.80
128(4, 4) + MP (4, 4)	32(3, 3) + MP (3, 3)	64 (2, 2) + MP (2, 2)	-	64 w/ 0.3 dropout	10	-	0.79
128(4, 4) + MP (4, 4) w/ 0.3 dropout	64(3, 3) + MP (3, 3)	32 (2, 2) + MP (2, 2)	-	128 w/ 0.3 dropout	10	-	0.84
128(4, 4) + MP (4, 4) w/ 0.3 dropout	64(3, 3) + MP (3, 3)	32 (2, 2) + MP (2, 2)	-	128 w/ 0.3 dropout	64 w/ 0.3 dropout	10	**0.85**
128(4, 4) + MP (4, 4) w/ 0.3 dropout	64(3, 3) + MP (3, 3)	32 (2, 2) + MP (2, 2)	-	64 w/ 0.3 dropout	128 w/ 0.3 dropout	10	0.84

1 - Changing dataset & Architecture

Confusion matrix and history for the best performing model:





2 - Different Features

I tried experimentally to add more features to the input apart from the already considered MFCCs.

I tried to use only the **Chroma Features** with the best performing model. The best result in accuracy was: **0.55**

I tried to consider (all together):

- **MFCCs**
- **Chroma features**
- **Spectral centroid** ➜ where the center of mass of the spectrum is located
- **Spectral contrast** ➜ considers the spectral peak, the spectral valley, and their difference in each frequency subband

The best result (overall) in accuracy was: **0.69**

3 - Trying an RNN approach

I tried to completely change the approach, passing from a CNN architecture to a RNN approach.

In particular, I have used LSTM layers, since I’ve found projects on the internet documenting their good performances for genre recognition. Despite this, I didn’t find any well documented paper with code related to the use of this architecture.

So I made some trials using an LSTM approach, using the data retrieved for the former experiments.

3 - Trying an RNN approach (2)

For this experiments I have organized the network as follows:

- GTZAN dataset
- 6 sequential layers:
  - 1 LSTM layer with 128 neurons, return_sequences=True
  - 1 LSTM layer with 64 neurons
  - 1 fully connected layer with 64 neurons➜ activation = “relu”
  - 1 fully connected layer with 10 neurons➜ activation = “softmax”
- optimizer ➜ adam
- input shape ➜ np.array with shape (n_segments, num_mfcc_vector_per_segment, mfcc_components) ➜ one dimension less with respect to the CNN method

3 - Trying an RNN approach (2)

Starting from this basic architecture I tried to find out which of the previously shown data worked better:

n_stft	hop_lenght	mel_components	val_accuracy
2048	1024	128	**0.87**
2048	1024	40	0.85
2048	1024	13	0.83
1024	1024	128	0.86
1024	1024	40	0.85
2048	512	40	0.84
2048	512	13	0.83
1024	512	40	0.84
512	512	40	0.83
512	256	40	0.80

3 - Trying an RNN approach (2)

I made also some experiments (as I did for the CNN) with **Chroma Features** and **Chroma Features, MFCCs, Spectral centroid and Spectral contrast**. This were the best results in accuracy:

- Chroma Features ➜ **0.48**
- Ch + MFCC + SpCent + SpCont ➜ **0.74**

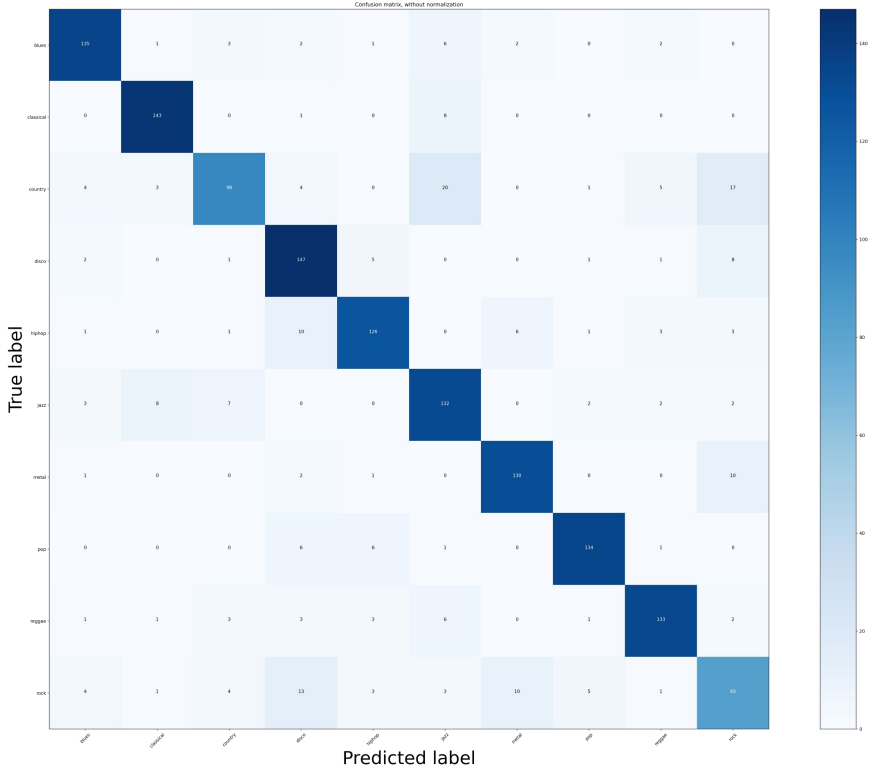
3 - Trying an RNN approach (2)

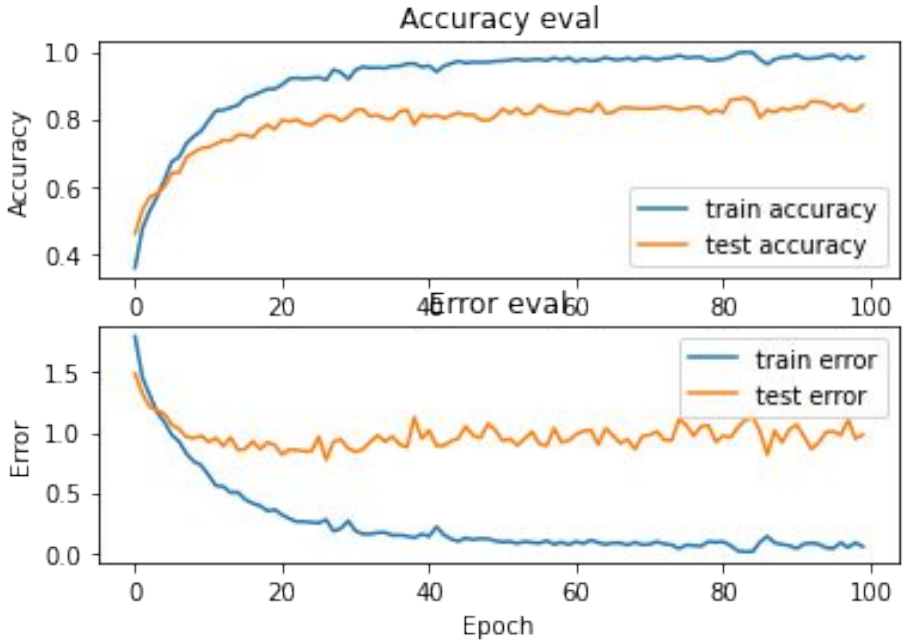
I played with the best performing data, trying to modify the network. I make here some examples:

LSTM_1	LSTM_2	LSTM_3	DENSE_1	DENSE_2	DENSE_3	**BEST RESULT**
128 (dropout 0.05)	64 (dropout 0.05)	32 (dropout 0.05)	64 (dropout 0.3)	10	-	0.86
128 (dropout 0.1)	64 (dropout 0.1)	-	128 (dropout 0.3)	10	-	0.85
64 (dropout 0.2)	32 (dropout 0.2)	-	128 (dropout 0.3)	10	-	0.83
128 (dropout 0.05)	64 (dropout 0.05)	32 (dropout 0.05)	128 (dropout 0.3)	64 (dropout 0.3)	10	**0.88**
128 (dropout 0.05)	64 (dropout 0.05)	32 (dropout 0.05)	64 (dropout 0.3)	32 (dropout 0.3)	10	0.86
64 (dropout 0.05)	64 (dropout 0.05)	-	128(dropout 0.3)	64(dropout 0.3)	10	0.84

3 - Trying an RNN approach (3)

Confusion matrix and history for the best performing model:





4 - Changing Scope ➔ From Genre to Artist Recognition

I tried to change the scope of the project: from a genre recognition to an artist recognition.

For doing so, I took into account another dataset: **artist20 dataset**, which is composed as follows:

- 1412 tracks, composed of six albums from each of 20 artists
  - ["aerosmith", "beatles", "creedence_clearwater_revival", "cure", "dave_matthews_band", "depeche_mode", "fleetwood_mac", "garth_brooks", "green_day", "led_zeppelin", "madonna", "metallica", "prince", "queen", "radiohead", "roxette", "steely_dan", "suzanne_vega", "tori_amos", "u2"]
- 22 kHz sample rate
- I took into account only the first 30 seconds of each song, for making it comparable with the previous experiments

4 - Changing Scope ➔ From Genre to Artist Recognition

The main problem with this dataset is the overfitting. Due to the low number of track per class this problem is very difficult to tackle.

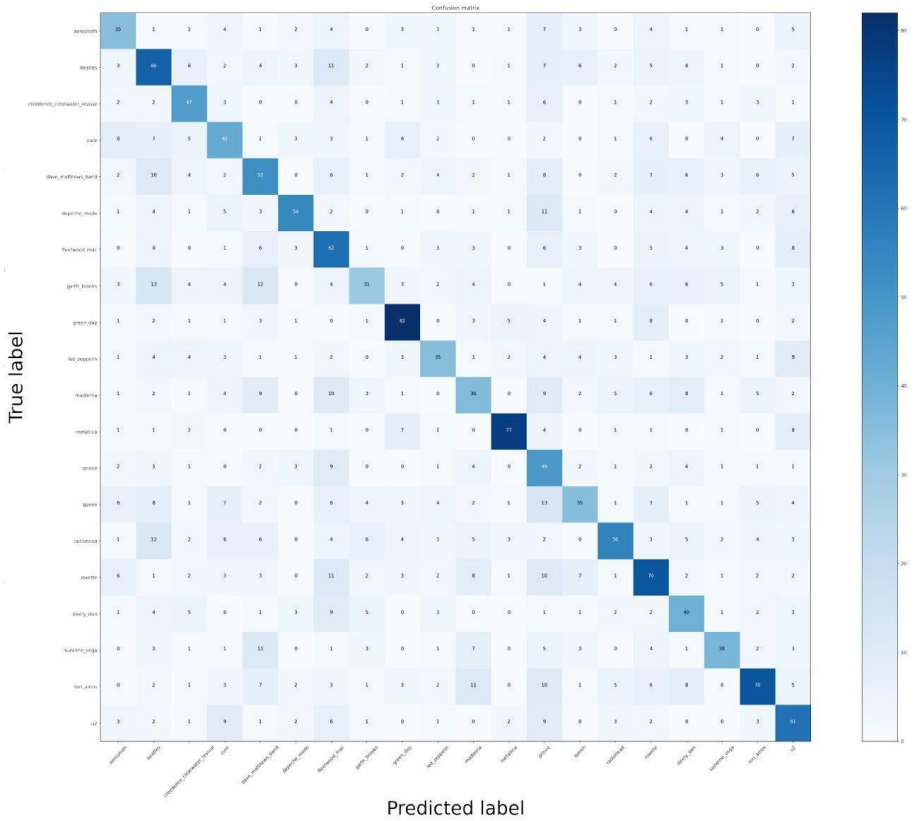
I made several experiments with different architectures and different parameters.

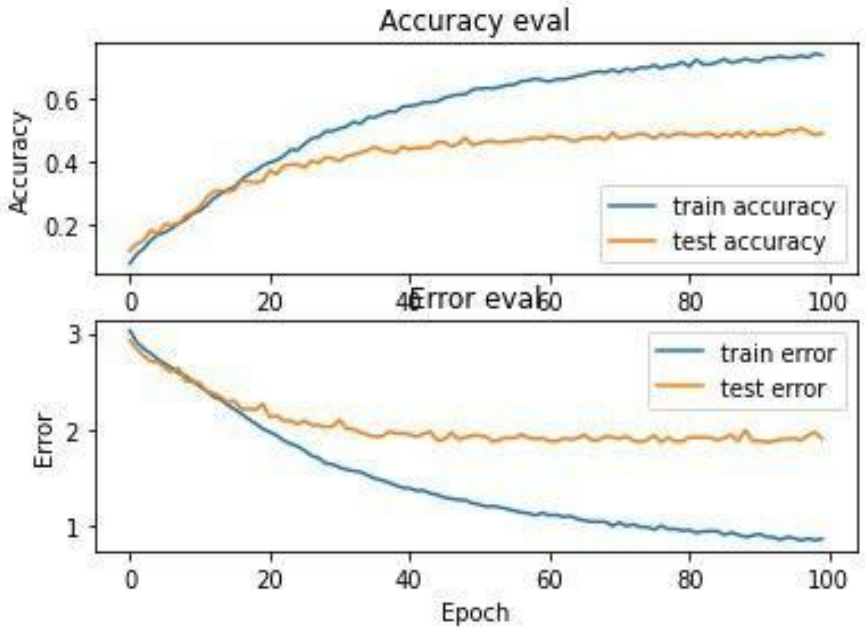
The best accuracy results obtained respectively with CNNs and LSTMs were:

- CNN - **0.44**
- LSTM - **0.54**

4 - Artist Recognition

CNN - 128(4, 4) + MP (4, 4) w/ 0.3 dropout

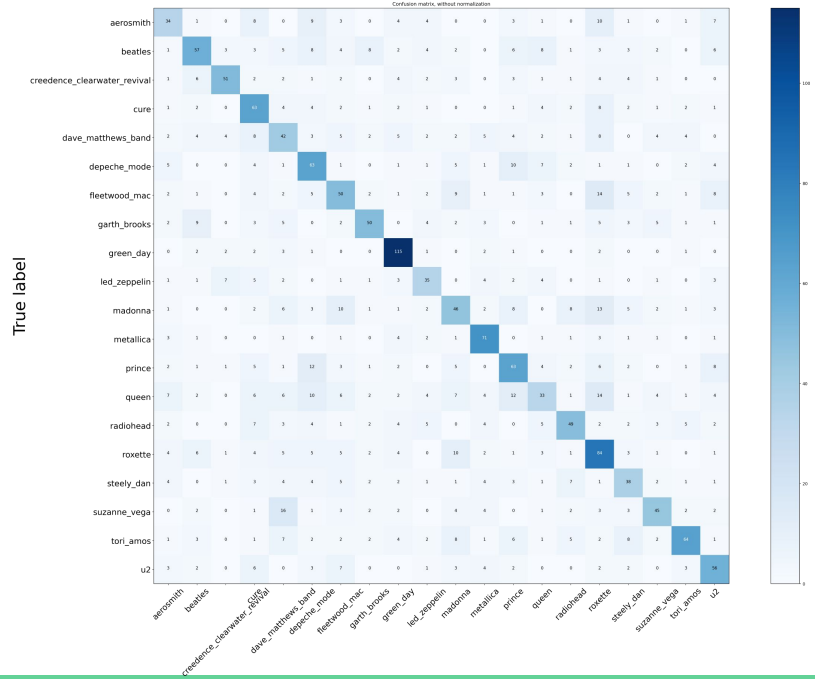


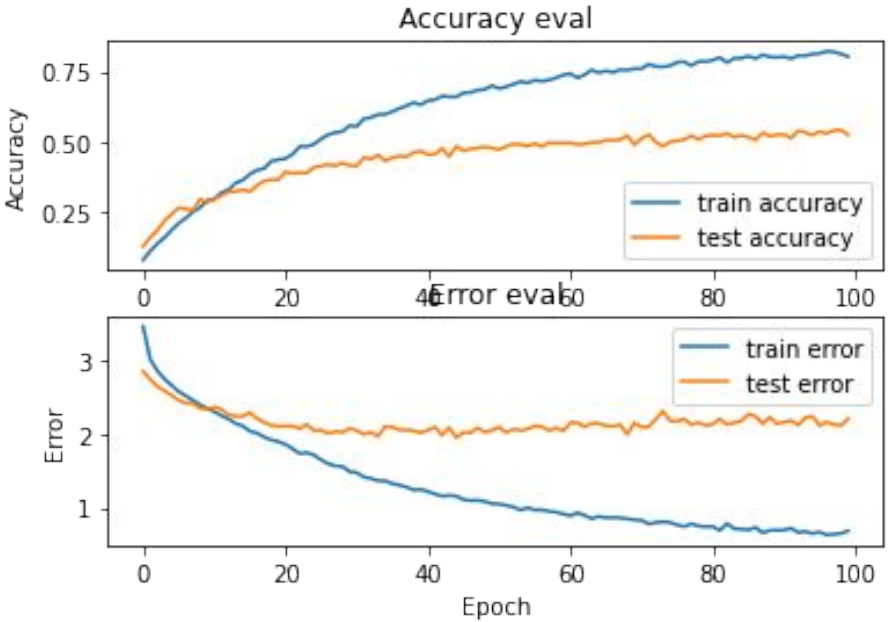


4 - Artist Recognition - LSTM

LSTM ➜ best results obtained with the following architecture:

LSTM 128 (drop 0.2) ➜ **LSTM** 64 (drop 0.2) ➜ **Dense** 128 (drop 0.4) ➜ **Dense** 64 (drop 0.4) ➜ **Dense** 10





4 - Artist recognition (Another dataset)

In the end I tried to compare the performance of this exact model with a custom dataset I created (for personal interest and for fun, of course not scientifically relevant).

This dataset is composed by 1400 tracks from 10 different italian pop artists:

[“articolo_31", "battiato", "de_andre", "elio_e_le_storie_tese", "giorgia", "guccini", "ligabue", "pooh", "vasco_rossi", "zucchero"].

4 - Artist recognition (Another dataset)

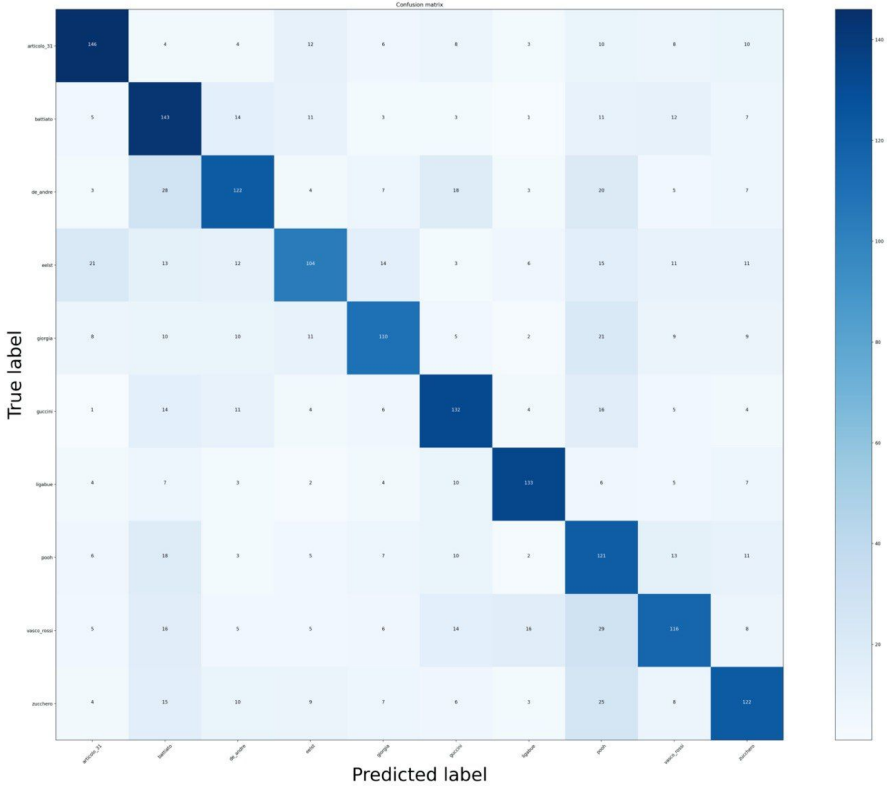
Again, here the similarities between the different artists are several and recurrent, but the increase in data size (same number of tracks but with half the classes) brought with it an increase in performance. These are the results performed on the exact same models as before:

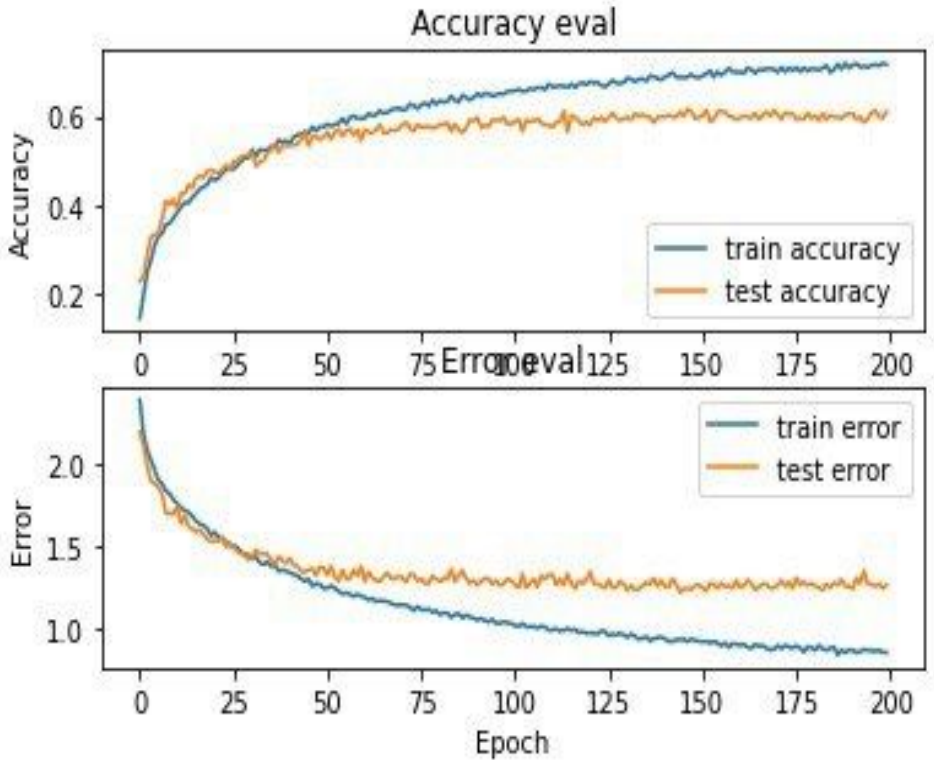
CNN ➜ **0.62**

LSTM ➜ **0.59**

4 - Artist recognition (Another dataset)

Confusion matrix and history for the best performing model:





5 - Conclusions

In this project I have analyzed a few ways for classifying music genre using neural networks, focussing on an end-to-end approach and starting from row-data.

The experiments I carried on showed that the differences among different MFCCs parameters for this task are minimal. At the same time I confirmed that the use of parameters usually worse the network performances.

CNNs and LSTMs networks perform comparably good, obtaining results not far from the current state-of-the-art, while having some problems with artist classification, since there the differences are less pronounced.