MACHINE LEARNING PROJECT

Music Genre Recognition with Neural Networks

Professor: *Andrea Asperti*
Submitted by: *Andrea Poltronieri*

TOPIC AND OBJECTIVES

In this project I have tried to explore different approaches to Music Genre Recognition, a field of research belonging to MIR (Music Information Retrieval). In particular, I started from a standard CNN approach, and then I made some experiments with RNN.

In this resume is aimed at explaining the structure of the project and the workflow I have used, showing some of the results retrieved.

Summarizing, these are the approaches I tackle in this project:

1. how to deal with a small dataset
2. music genre recognition using **CNN**
    a. data parametrization
        i. MFCCs
            1. what's the best set of parameters to be used with a small dataset and CNN?
        ii. Chroma features
        iii. Spectral centroid
        iv. Spectral contrast
    b. setting the network for obtaining the best result
3. music genre recognition using **RNN** (LSTM)
    a. data parameterization (see above)
    b. setting the network for obtaining the best result
4. changing the project's scope: from genre to **artist recognition**
    a. experiments with CNN and LSTM
    b. experiments with a custom dataset
5. Conclusions

BIBLIOGRAPHIC REFERENCES

Before I started coding for the project I read several papers about MIR and, in particular, about classification tasks.
I came up with a lot of material, from which I decided to start from a "flat" approach which could have provided me with the possibility to explore other architectures and data features. The paper I started the project from was:

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

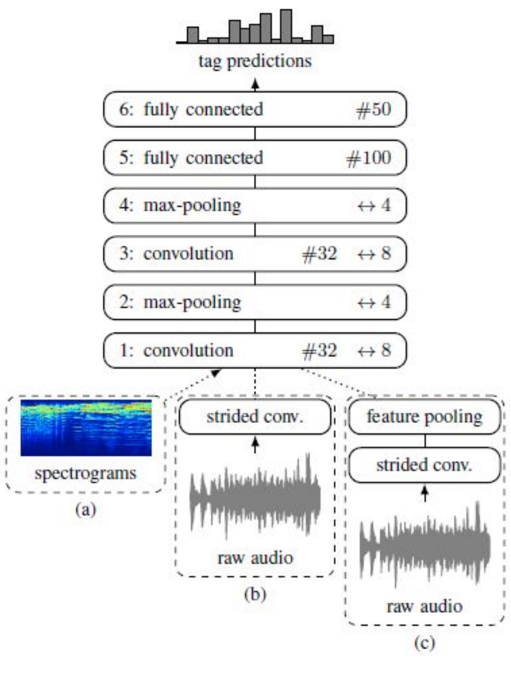
For the project section related to RNN I couldn't find any well-structured scientific paper on the theme, but I consulted different MIR experts' repositories aimed at solving the same problem I was dealing with.
For the last section of the project, the one related to artist recognition I mainly referred to:

> *Nasrullah, Z. and Zhao, Y., "Musical Artist Classification with Convolutional Recurrent Neural Networks," International Joint Conference on Neural Networks (IJCNN)*

1. CHANGING THE DATASET

In the paper, the authors explore music classification by using CNN. They used a fairly big dataset (**Magnatune**: 25863 29-seconds audio clips, 16 kHz sample rate, annotated with 188 tags, where only the main 50 have been considered), which have been the first thing I decided to change.
I decided to use a broad-used dataset for music recognition, but, on the contrary, pretty small: **Marsaya's GTZAN**: 1000 30-seconds clips, 22 kHz sample rate, annotated with 10 tags.

Initially, I had to deal with the small dimensions of the dataset I decided to use. To overcome this problem I split every track of the dataset in 10 segments. Then, I calculated the MFCCs vectors for each segment, which have been associated with 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 *“End-to-end learning for music audio”* paper is performing better, without any normalization. As proposed in this 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.

Then I applied the same architecture proposed in the aforementioned paper to GTZAN dataset and I compared the results for each set of MFCCs parameters extracted.
The starting architecture was the following:

- 6 sequential layers (details in the image of the previous page):
    - 1 conv2D layer + 1 max pooling
    - 1 conv2D layer + 1 max pooling
    - 1 fully connected layer
    - 1 fully connected layer
- optimizer ➔ adam
- loss ➔ categorical cross-entropy
- input shape ➔ np.array with shape (n_segments, num_mfcc_vector_per_segment, mfcc_components, channel [=1])

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 the Keras library.

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

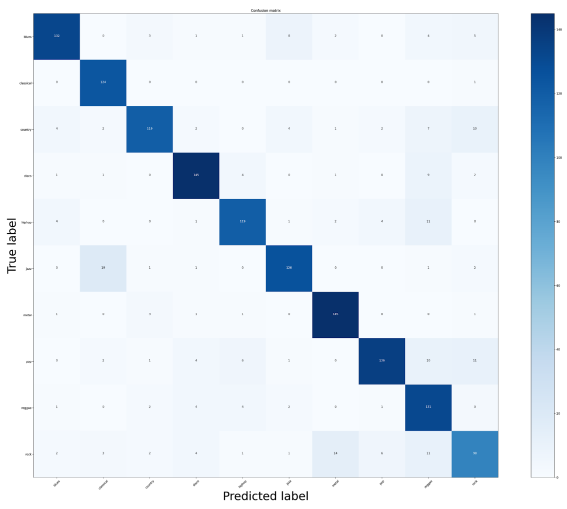
Here I propose a small example of the accuracy performances of the original model on GTZAN dataset, considering different sets of MFCCs parameters:

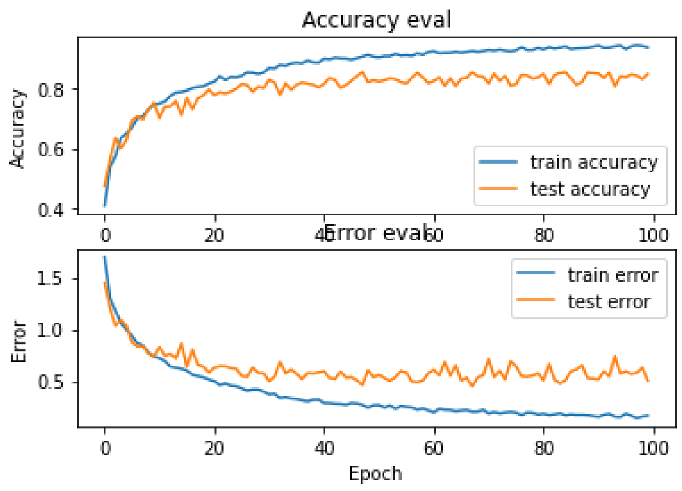
n_stft	**hop_lenght**	**mel_components**	**val_accuracy**
2048	1024	128	0.56
2048	1024	40	**0.64**
2048	1024	13	/

1024	1024	128	0.38
1024	1024	40	0.49
2048	512	40	0.31
2048	512	13	/
1024	512	40	0.30
512	512	40	0.52
512	256	40	0.29

Then, starting for the best performing sets of MFCCs parameters, I played with the architecture of the CNN, until I obtained a result of accuracy comparable with the one proposed in the reference paper (0.88): **0.85**.
Here I propose some of the architectures tested:

CNN_1	CNN_2	CNN_3	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)	64w/ 0.3 dropout	10	-	0.79
64(4, 4) + MP (4, 4)	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)	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)	64(3, 3) + MP (3, 3)	32 (2, 2) + MP (2, 2)	64 w/ 0.3 dropout	128w/ 0.3 dropout	10	0.84





I made some experiments also with other parameters extracted from the dataset, always obtaining worse results:

- Croma features ➔ **0.55** accuracy
- MFCCs+Chroma features+Spectral centroid+Spectral contrast ➔ **0.69** accuracy

2. TRYING WITH A RNN APPROACH

I tried to completely change the approach, passing from a CNN architecture to an RNN approach. In particular, I have used LSTM layers, since I've found projects on the internet documenting their good performances for genre recognition. Then I made some trials using an LSTM approach, using the data retrieved for the former experiments.

I started from a quite simple LSTM net for comparing the data features (as well as in the CNN workflow):

- 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
    - 1 fully connected layer with 10 neurons
- optimizer ➔ adam
- loss ➔ categorical cross-entropy
- input shape ➔ np.array with shape (n_segments, num_mfcc_vector_per_segment, mfcc_components) ➔ one dimension less with respect to the CNNs

Here I adopted the very same approach I adopted in the previous section:

- I tried to understand the best performing set of parameters to extract MFCCs

n_stft	**hop_lenght**	**mel_components**	**val_accuracy**
2048	1024	128	**0.87**
2048	1024	40	0.85
2048	1024	13	0.83

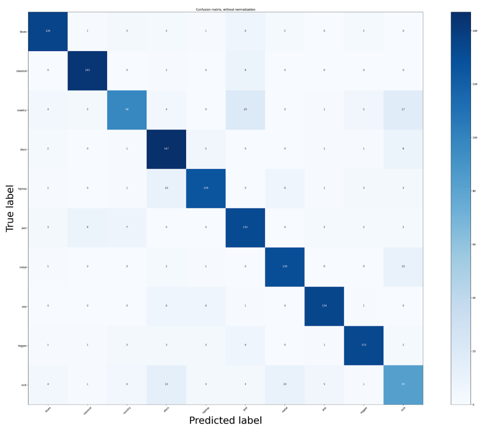
1024	1024	128	0.86
1024	1024	40	0.85
2048	512	40	0.84
2048	512	13	0.83
1024	512	40	0.84
512	512	40	0.83
512	256	40	0.80

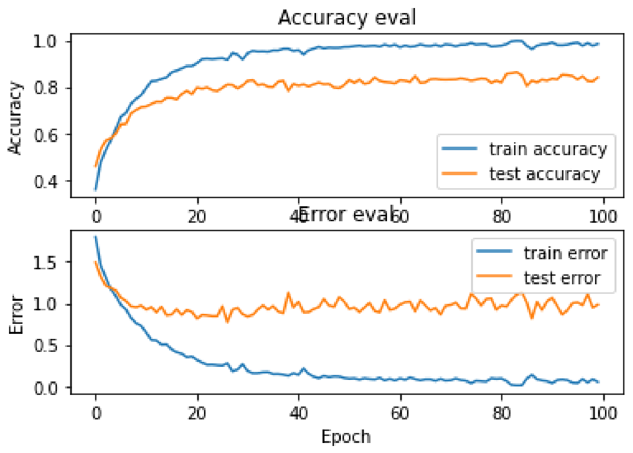
- I changed several times the LSTM-based architecture, in order to obtain better results:

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

- I tried to use the same network with Chroma features and the other features aforementioned (respectively **0.48** and **0.78** accuracy).

Here the best result obtained was **0.88** accuracy, which is plotted below.





3. CHANGING THE PROJECT'S SCOPE

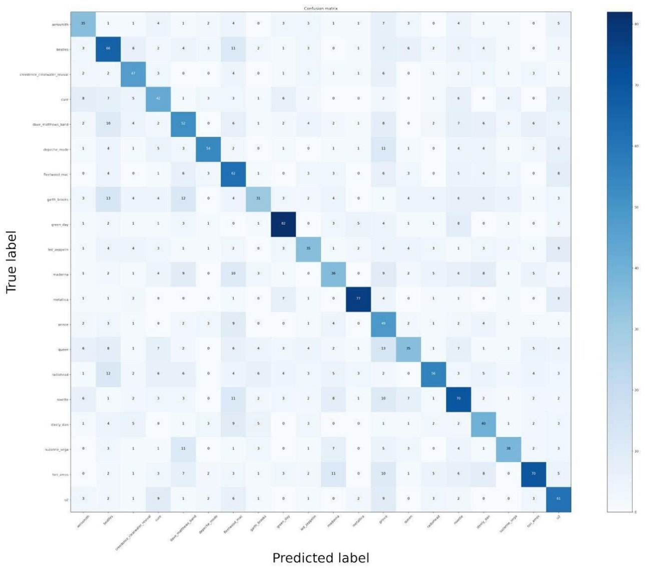
In the last part I tried to change the scope of the project: from a genre recognition to an artist recognition.
For doing so, I took into account another dataset: **artist20 dataset**, which is composed as follows:

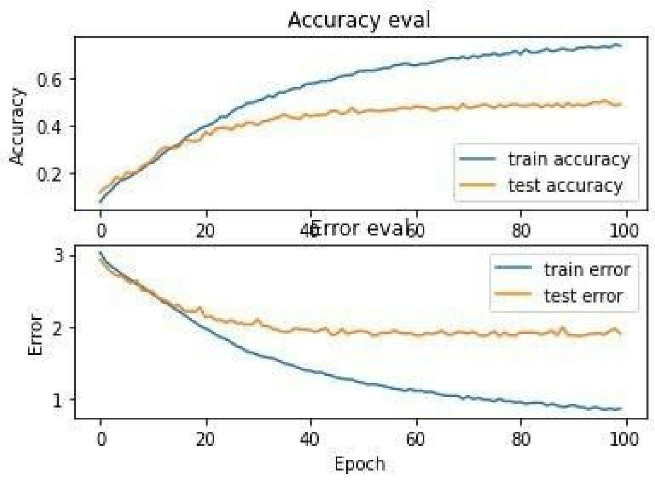
- 1412 tracks, composed of six albums from each of 20 artists
  ["aerosmith", "beatles", "creedence_clearwater_revival", "cure", "dave_matthews_band", "depeche_mode", "fleetwood_mac", "garth_brooks", "green_day", "led_zeppelin", "madonna", "metallica", "prince", "queen", "radiohead", "roxette", "steely_dan", "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

The main problem with this dataset is the overfitting. Due to the low number of tracks per class this problem is very difficult to tackle, also considering the genre similarities of most of the dataset's tracks.

I made several experiments with different architectures and different parameters. These are the best accuracy results obtained respectively with **CNNs** and **LSTMs**:

- CNN ➔ **0.44**
- LSTM ➔ **0.54**

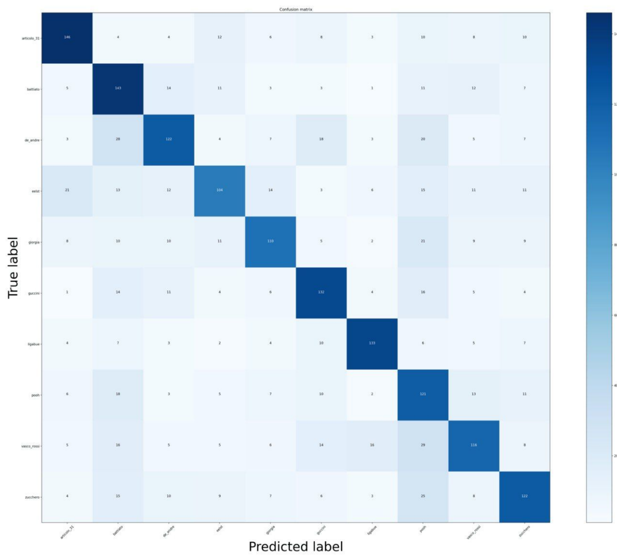


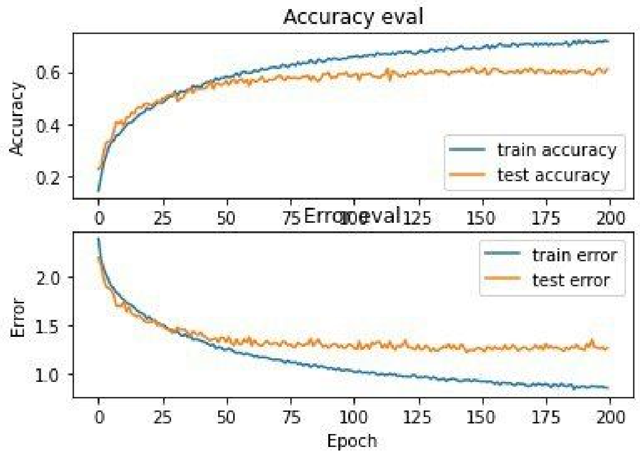


In the end I tried to compare the performance of this exact model with a custom dataset I created (for personal interest and for fun, of course not scientifically relevant).

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"].
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**





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.