Music Genre Classification

Andrea Trianni, Chiara Chiucchi, Emanuele Mercanti, Matteo Pascolini

Contents

1 Abstract

In the following work, we decide to solve the task of **Music Genre Classification** with different models. Being able to classify an audio track for its author, genre, etc. it is a useful task for many Recommendation System (i.e. to suggest songs).

In this report we are going to talk about:

- **How to work with audio data:** what is audio, how it is represented, how to extract the meaningful features of audio.
- **Related Work:** research in Music Genre classification and its most common solutions.
- **Proposed Method:** models used to solve the Music Genre Classification task, why they are used and how they are used.
- **Dataset and Benchmark:** the dataset used for the evaluation, its faults and resulting modifications; the referred benchmark and how it is obtained.
- **Experimental Results:** models' evaluations with different figures of merit and considerations.
- **Conclusions and Future Work:** faults of the proposed solution and possible other modifications.

2 Introduction

In this section we are going to talk about **audio** data, without going to much in details but exploiting the main concepts needed to understand our task.

A **sound** is produced when an object vibrates, determining the oscillation of molecules; such oscillations produce pressures, that in alternation with each other create a **wave**.

Basically a sound can be represented by a **waveform**:

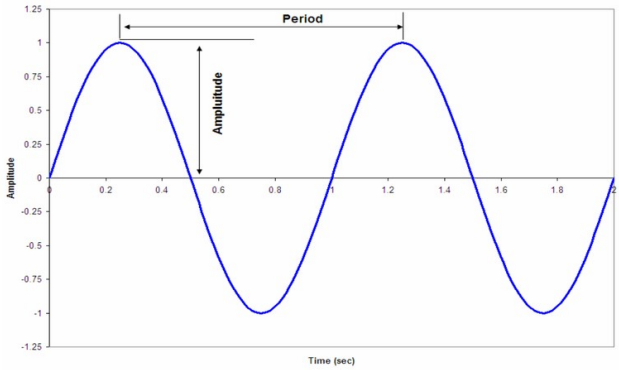


Figure 1: Example of a sound **waveform**, with its **amplitude** and **period**

As we can see in figure 1, a **wave** can be represent by its **amplitude** over time: it can be seen as a point that oscillates with different amplitudes at different time. Another important is the **period**: it can be seen as the interval between the "starts" of the same wave (e.g. in figure 1 the period is the interval between two peaks of the waveform). The period is strictly correlated with the **frequency**, indeed:

$$f = \frac{1}{T}$$

where T is the period.

2.1 Digitize Audio

When we are considering **acoustic sound**, like the sound of a piano or an instrument in general, we are talking about *continuous* or *analog* waveform. One possible procedure to digitize an analog waveform is the **Analog Digital Conversion** (ADC), which first samples the signal at uniform time intervals (*sampling rate*) and then quantizes the amplitude with limited number of bits (*bit depth*). For example with CD-ROM we have sample rate of 44.100 Hz (samples per second) and a bit depth of 16 bits per channel.

When considering real-world sound waves it turns out that their waveforms are a little bit more complex for this kind of digital conversion. However nature gives us tools to extract very useful features for audio.

2.2 Fourier Transform (FFT)

Let's take as example one of the audio track in the dataset that we are going to use after.

As we can see in figure 2 the wave is pretty complex. However, we can know more about complex sounds thanks to the **Fourier Transform** which decomposes complex periodic sound into a sum (**superposition**) of sine waves oscillating at *different frequencies* and *different amplitudes*.

Applying a Fourier Transform on the wave we can obtain a **Power Spectrum** that gives as the **magnitude** (power) as function of the frequency. In figure 3 it is represented the Power Spectrum of the waveform considered in figure 2.

We can see that most of the energy is concentrated in the *lower frequencies* and that as the frequency increases, the power goes lower and lower. When we apply the FFT (Fast Fourier Transform):

- We are moving from the **time domain** to the **frequency domain**.
- So, there is **no time information**.

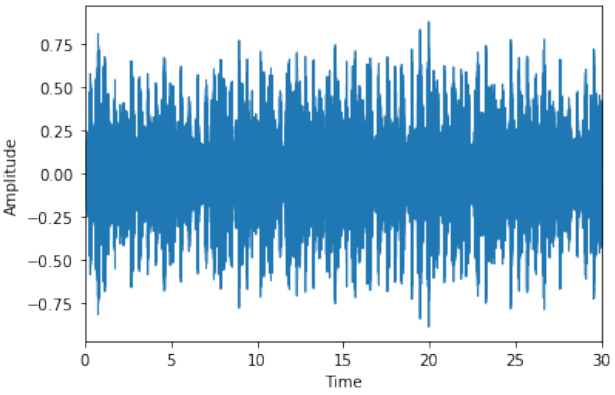


Figure 2: **Waveform** of the `blues.00000.wav` audio file.

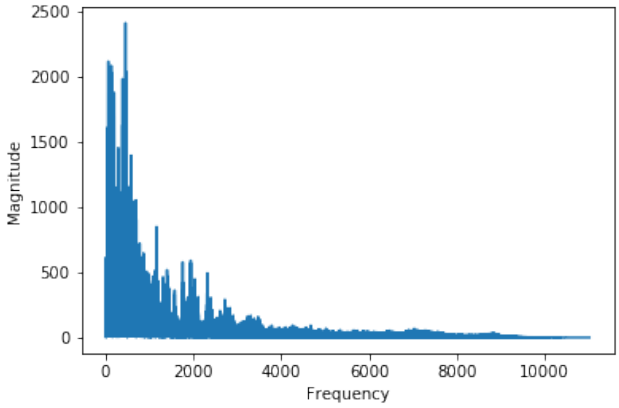


Figure 3: **Power Sprectrum** of the `blues.00000.wav` audio file.

It is clear how the STFT gives more information about audio data wrt the FFT. But there exists another feature that is more important and detailed than STFT.

2.4 Mel Frequency Cepstral Coefficients (MFCCs)

The **MFCC**s are frequency domain features able to capture timbral/textural aspects of sound e.g. distinguish between a piano and a violin playing the same melody. The great advantage of the MFCCs is that they approximate the human auditory system: that's why they are one of the most used features in Music Identification/Recognition tasks.

Basically an MFCC is a vector of **coefficients**: usually in audio application the number varies from 13 to 40 coefficients. These coefficients are calculated at each

Then, we need a way to preserve time information.

2.3 Short Time Fourier Transform (STFT)

The **STFT** (Short Time Fourier Transform) computes several FFT at different intervals, preserving time information. The different intervals are given by a fixed **frame size** (e.g. 2048 samples). This procedure gives us a **spectrogram** which represents the magnitude as a function of both frequency and time.

Figure 4 shows the **spectrogram** of our considered sample. We can see that most of the frequencies contributes very little to the overall sound. While we can see more contribution at *lower frequencies*, which was what we were expecting from the **power spectrum** in figure 3.

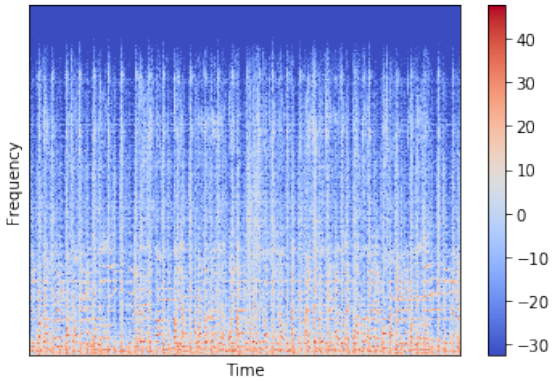


Figure 4: **Spectrogram** of the `blues.00000.wav` audio file.

frame, so that we can have an idea of how MFCCs evolve over time.

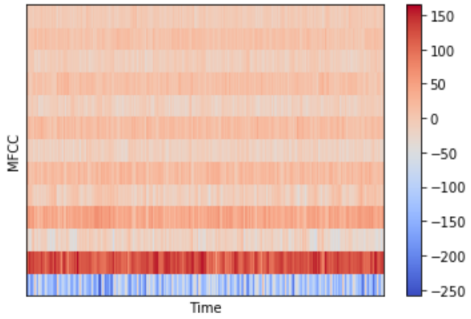


Figure 5: **MFCC**s of the `blues.00000.wav` audio file.

Figure 5 shows the MFCCs of our sample. It is clear that the result is very similar to the one in figure 4 and it can be read in the same way (more power on lower frequencies). Our work uses 13 MFCCs for each audio track as features to train/test the models to solve the task of Music Genre Classification.

3 Related Work

For the preprocessing part, we have used the ideas and part of the code suggested by [5]
The architectures, they were either inspired by the [5] playlist or by research papers further explaination is provided into the proposed methods section.

4 Proposed Method

4.1 Multi-layer Perceptron Network

The simpler model we have used is a multi layer perceptron (MLP). The model has undergone many changes in order to improve its performance.
The first MLP architecture is composed by 5 fully connected layers, each activated by a ReLU, and a final softmax activation layer. The first layer has 500 units and each subsequent layer has less units than the previous one as shown in **Figure 6**.

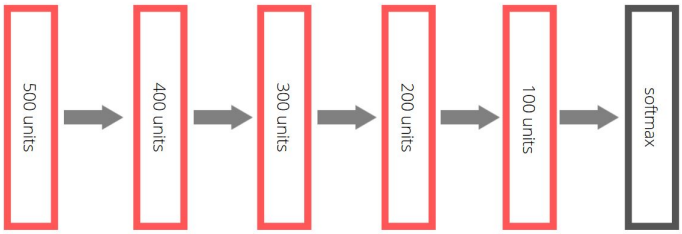


Figure 6: Initial architecture

This way the model progressively encodes information, reducing the importance of noise and high level features, high pitch tones and notes such as guitar. This is useful because the low level features, such as the drum rhythm and the bass guitar rhythm usually define the genre of a song: different genres have different drum and bass patterns.
Moreover, we use the Adam optimizer with a learning rate of 10^{-5}. This model is prone to over-fitting and performs poorly, reaching an accuracy on test set of approximately 60% (**Figure 7**).

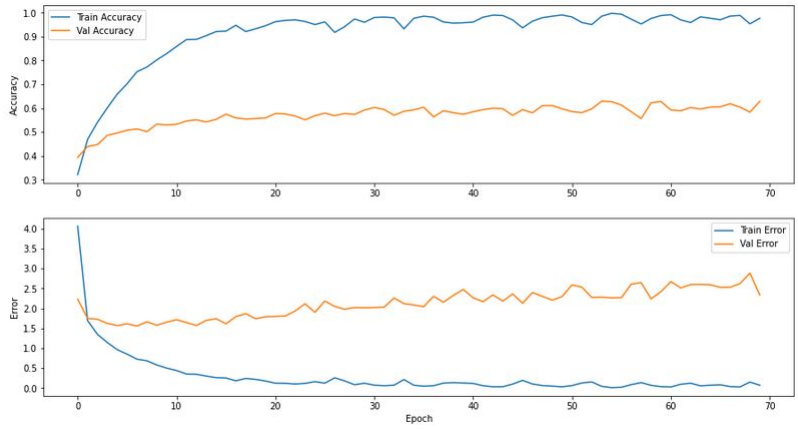


Figure 7: Initial architecture performance

To reduce the over-fitting problem and to improve performance some changes are made. First of all early stopping is added: if the validation set loss doesn't drop in 10 epochs the model stops the training. Then the L2 kernel regularizer is implemented to reduce the importance of useless features. Lastly, dropout is added. Concerning the dropout, three different settings were tried:

1. Equally drop 30% of units in every layer;
2. Drop more units in early layers: 50% of units in layer 1 and 2, 25% of units in layer 3 and 4, 10% of units in layer 5;
3. Drop less units in early layers: 10% of units in layer 1, 25% of units in layer 2 and 3, 50% of units in layer 4 and 5.

The best dropout rate is the 3^{rd} one. We believe this is related to the early layers learning low level features (bass and drums) which are more important as mentioned earlier. These changes results in results in highly reducing overfitting, however there is a little gain in accuracy. The accuracy reached by this optimized model is approximately 62%. To further improve the model accuracy more changes are introduced. The Exponential Linear Unit (ELU) activation function was used instead of the ReLU, as suggested in the paper [2].

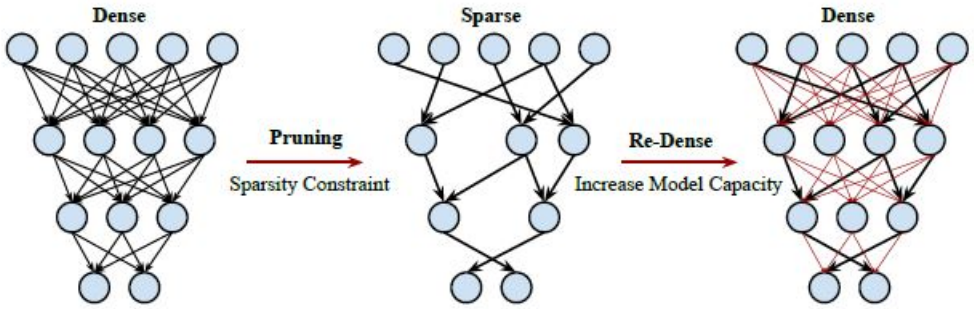


Figure 8: Dense Sparse Dense Training: Dense-Sparse-Dense Training Flow. The sparse training regularizes the model, and the final dense training restores the pruned weights (red), increasing the model capacity without over-fitting.

Lastly, a Dense-Sparse-Dense (DSD) training is implemented, as presented in [4]. The DSD architecture is shown in implemented in **Figure 8**. The Dense-Sparse-Dense training consists of training the neural network one first time (Dense). Then training it a second time using the weights of the first training, and filtering out the weights that are too close to zero (Sparse). Lastly the network is trained a third time using the weights of the second training, with restored connections (Dense). As suggested in the paper, we use a sparsity of 30%.
Quoting the paper from which the architecture of the MLP is defined [4]: *The dense-sparse-dense training*

framework that regularizes neural networks by pruning and then restoring connections. This method learns which connections are important during the initial dense solution. Then it regularizes the network by pruning the unimportant connections and retraining to a sparser and more robust solution with same or better accuracy. Finally, the pruned connections are restored and the entire network is retrained again. This increases the dimensionality of parameters, and thus model capacity, from the sparser model.

4.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are mostly used in image and video recognition but they have been also actively used for various music classification tasks such as music tagging and genre classification, because the audio features extractable from audio files, **MFCCs** and **spectograms** can be seen as images.

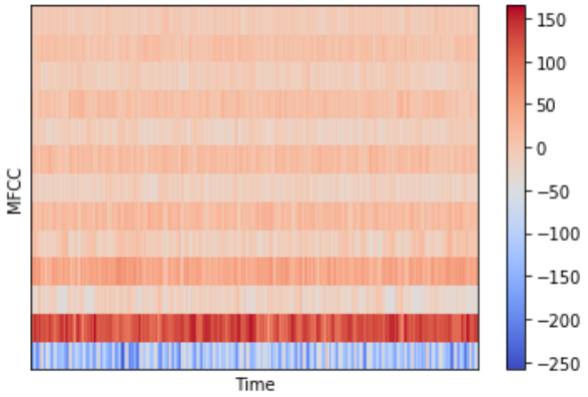


Figure 9: The structure of an image that represents an MFCC

Our model is composed by three hidden layers and a dense layer. Each hidden layer is composed by a **2D Convolution** operation followed by a **Max Pooling** operation. In all 2D Convolution operations the number of filters (32) and the activation function (ReLU) never change, whereas the kernel dimension changes in the last hidden layer: (3 x 3) for the first and second layer, (2 x 2) for the last one. Similar situation in Max Pooling operations where a (3 x 3) and (2 x 2) pool windows are used: the former for the first and the second hidden layer, the latter for the third layer. The pool window shifts by 2 for each dimension at each step in every hidden layer. The final layer assigns probabilities to each class with a softmax function. We carried out the hyperparameter tuning process in order to reach the best values for our model. As optimizer, we use the Adam optimizer with a learning rate of 10^{-5} and a 20% of unites dropped in order to reduce overfitting.

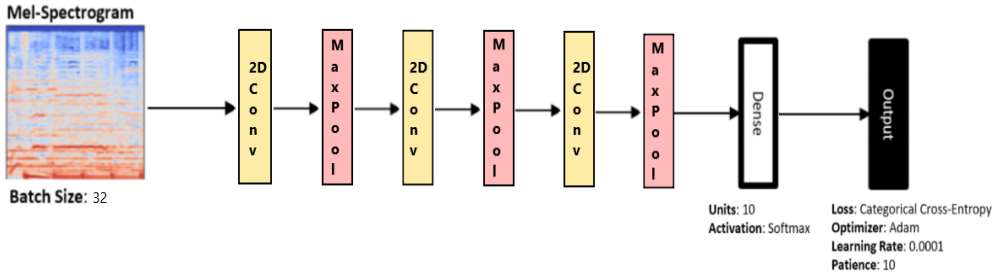


Figure 10: CNN architecture for Music Genre Classification

4.3 Recurrent Neural Network

Recurrent Neural Networks (RNNs) are deep learning models very powerful to recognize patterns in signals that evolve over a sliced window time. Since music is sound and this last is a signal that evolves over time, this architecture seems to be particularly suitable for this kind of problem.

Since it is not the purpose of this document to explain the mathematical formalism related to recurrent models, we will avoid further considerations related to formulas modeled by these neural networks, and we will keep the focus on the work we have done.

Speaking about our particular model, it involves the use of LSTM (Long Short Term Memory) cells, instead of the vanilla standard solution or the GRU cells. Our choice was dictated by the fact that these cells are very powerful and solve the vanishing gradient problem. As you can see form **Figure 11**, our model consist of two sequential lstm cell with hidden matrix size of 64, and two final dense layer with softmax function at the end. Between the layer and the cells, dropout is widely used to prevent overfitting, and Relu non-linear activation function is used

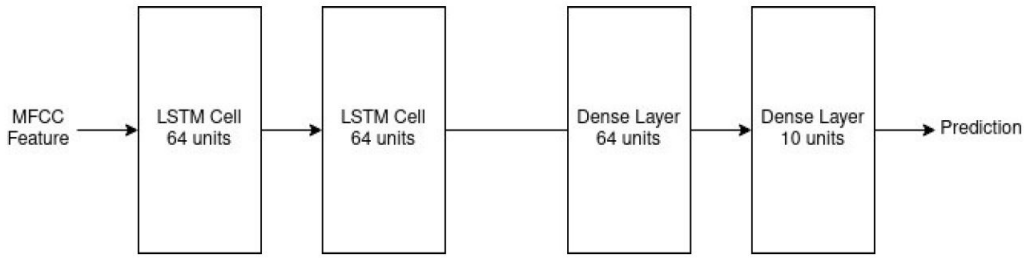


Figure 11: RNN architecture for Music Genre Classification

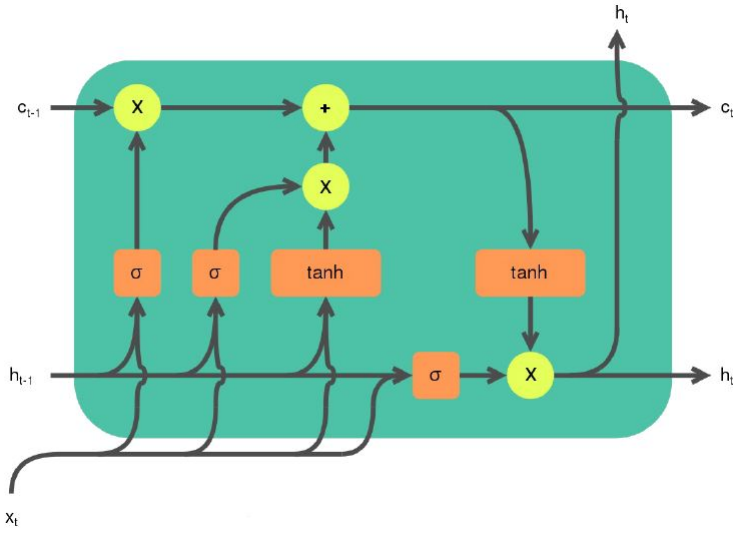


Figure 12: View of the lstm cell, and its gates, seen from the inside

As with the other models, the hyper parameters of this model were found experimentally and manually. Since this architecture, more than the others, requires a lot of computational effort and time to train, we have decided not to go beyond 0.001 of learning rate. Other hyper parameters, such as batch size, remained unchanged respect the previous models.

4.4 Convolutional Recurrent Neural Network

The ideal network to perform Music Genre Classification, should be able to summarize patterns in frequency (where convolutional layers excel) and then also consider any resulting temporal sequences in these patterns (where recurrent layers excel).

Indeed, *Convolutional Neural Network* act as feature extractor: low level layers describe sound onsets or the presence of individual instruments, while high-level layers describe abstract patterns. Also *Recurrent Neural Network* have also had success with audio tasks, because sound is sequential along the time axis. **The Convolutional Recurrent Neural Network** contains both of these components.

The final model is described and used both in the Choi [2] and Zhao [3] works: the first demonstrates how the CRNN performs better against three different models of CNNs; the second work better specifies the CRNN architecture - which is the same of the first paper even if it is performed a different classification task.

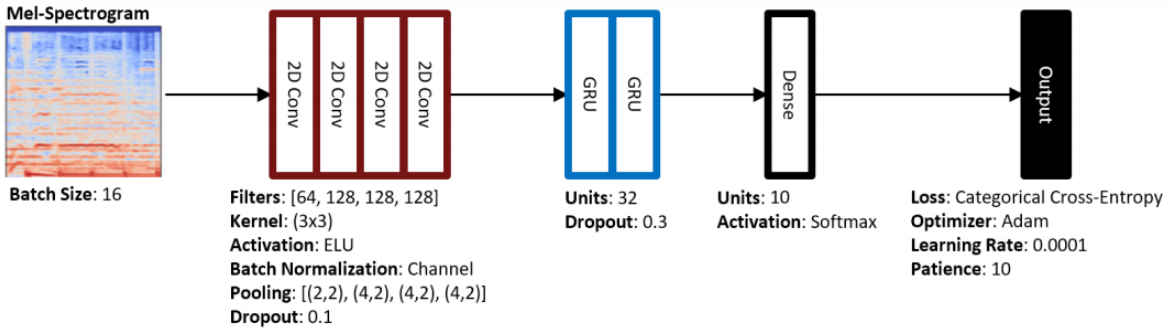


Figure 13: CRNN architecture for Music Genre Classification.

Figure 13 shows the architecture of the CRNN, that can be described by three stages: *convolutional*, *recurrent* and *fully-connected*.The *Exponential Linear Unit* (ELU) activation function is used as a smooth alternative of the *Rectified Linear Unit* (RELU) because it has shown better generalization performance.

Batch Normalization (normalizing across channels) and *drop-out* (regularization) are also included to improve generalization. Then, *Gated Recurrent Units* (GRUs) are used for temporal summarization, instead of *Long Short Term Memory* (LSTM) cells, since they require fewer parameters and have similar performance. The final fully-connected layer assigns probabilities to each class with a softmax function.

5 Dataset and Benchmark

In this section we are going to talk about the used dataset for training and evaluation of the proposed models, its faults and how we decide to fix them. Then we are going to show the chosen benchmark and how it was obtained.

5.1 GTZAN Dataset

We decide to test our work using the **GTZAN dataset**. Our evaluation about this dataset comes from the work of Bob L. Sturm [1].

The GTZAN dataset is one of the most used for evaluation in machine learning for **music genre recognition** (MGR). It appears in at least 100 published works, but however, the research done by Sturm demonstrates that the GTZAN dataset has several faults, like *repetitions*, *mislabeling* and *distortions*.

5.1.1 Mislabeling

The author uses the **ENMFP** (Echo Nest Musical Fingerprint) platform to generate a *fingerprint* of every sample in GTZAN. Basically the platform maps an unknown audio file to a song represented with an ID and metadata such as the artist, the title, the genre and so on. The ENMPF was able to identify only 60.6% of the samples. The author manually identified some of the remaining unknown excerpts, missing only the identification of 81 samples.

The definition of **mislabeling** comes from the following reasoning: a music genre can be described as a set of **tags**; for example *"Reggae"* can be described by the tag *"110 bpm"* referring to its tempo, as *"Bob Marley"* referring to one of its possible artist, etc. For each genre, the author keeps track of how many previously identified samples can be described by the **top tags** of that genre; when tags for a song are not found, then the author gets the tags for the artist of that track.

Label	ENMFP	self	last.fm	
			song (no. tags)	artist (no. tags)
Blues	63	100	75 (2904)	25 (2061)
Classical	63	97	12 (400)	85 (4044)
Country	54	95	81 (2475)	13 (585)
Disco	52	90	82 (5041)	8 (194)
Hip hop	64	96	90 (6289)	5 (263)
Jazz	65	80	54 (1127)	26 (2102)
Metal	65	83	73 (5579)	10 (786)
Pop	59	96	89 (7173)	7 (665)
Reggae	54	82	73 (4352)	9 (616)
Rock	67	100	99 (7227)	1 (100)
Total	60.6%	91.9%	72.8% (42567)	18.9% (11416)

Figure 14: For each class of the GTZAN: number of samples identified by fingerprint (ENMFP); then searched manually (self); number of songs identified by tags (and number of tags); number of songs identified by tags of artist (and number of tags)

Given the results of figure 14, we can say for example that for the *"Country"* category at least 6 samples are mislabeled since those songs don't have enough properties correlated to that genre.

5.1.2 Repetitions

The paper considers four types of **repetition**:

1. **Exact repetition:** when two samples are the same such that their time-frequency fingerprints are highly similar.

   To find exact repetitions, the author implements a simple version of the *Shazam fingerprint* and he founds **50 exactly repetitions** in GTZAN.

2. **Recording repetition:** when two samples come from the same recording (not detected with fingerprinting).

   To find those the author checked for multiple artists and songs appearing multiple times in the data; then he further more check the repetition listening to those samples. Overall, the author found **21 recording repetitions** in GTZAN.

3. **Artist repetition:** samples performed by the same artist. Figure 15 shows how every category of GTZAN has artist repetition.

4. **Version repetition:** when two samples are of the same song but performed differently.

   As before, the author found these using the data and then confirming by listening; In total, he found **13 version repetitions** in GTZAN.

Label	Repetitions				Distortions
	Exact	*Recording*	*Artist*	*Version*	
Blues			John Lee Hooker (0-11); Robert Johnson (12-28); Kelly Joe Phelps (29-39); Stevie Ray Vaughn (40-49); Magic Slim (50-60); Clifton Chenier (61-72); Buckwheat Zydeco (73-84); Hot Toddy (85-97); Albert Collins (98, 99)		
Classical		(42,53) (51,80)	J. S. Bach (00-10); Mozart (11-29); Debussy (30-33); Ravel (34-37); Dutilleux (38-41); Schubert (63-67); Haydn (68-76); Grainger (82-88); Vivaldi (89-99); and others	(44,48)	static (49)
Country		(08,51) (52,60)	Willie Nelson (19,26,65-80); Vince Gill (50-64); Brad Paisley (81-93); George Strait (94-99); and others	(46,47)	static distortion (2)
Disco	(50,51,70) (55,60,89) (71,74) (98,99)	(38,78)	Gloria Gaynor (1,21, 38,78); Ottawan (17,24,44,45); The Gibson Brothers (22,28,30,35,37); KC and The Sunshine Band (49-51,70,71,73,74); ABBA (67,68,72); and others	(66,69)	clipping distortion (63)
Hip hop	(39,45) (76,78)	(01,42) (46,65) (47,67) (48,68) (49,69) (50,72)	Wu-Tang Clan (1,7,41,42); Beastie Boys (8-25); A Tribe Called Quest (46-51,62-75); Cypress Hill (55-61); Public Enemy (81-98); and others	(02,32)	clipping distortion (3,5); skip at start (38)
Jazz	(33,51) (34,53) (35,55) (36,58) (37,60) (38,62) (39,65) (40,67) (42,68) (43,69) (44,70) (45,71) (46,72)		James Carter (2-10); Joe Lovano (11-24); Branford Marsalis Trio (25-32); Coleman Hawkins (33-46,51,53,55,57, 58,60,62,65,67-72); Dexter Gordon (73-77); Miles Davis (87-92); Joe Henderson (94-99); and others		clipping distortion (52,54,66)

Label	Repetitions				Distortions
	Exact	*Recording*	*Artist*	*Version*	
Metal	(04,13) (34,94) (40,61) (41,62) (42,63) (43,64) (44,65) (45,66) (58) is Rock (16)		Dark Tranquillity (12-15); Dio (40-45,61-66); The New Bomb Turks (46-57); Queen (58-60); Metallica (33,38,73, 75,78,83,87); Iron Maiden (2,5,34,92-94); Rage Against the Machine (95-99); and others	(33,74) (85) is Ozzy Osbourne covering Disco (14)	clipping distortion (33,73,84)
Pop	(15,22) (30,31) (45,46) (47,80) (52,57) (54,60) (56,59) (67,71) (87,90)	(68,73) (15,21,22) (47,48,51) (52,54) (57,60)	Mandy Moore (00,87-96); Mariah Carey (2,97-99); Alanis Morissette (3-9); Celine Dion (11,39,40); Britney Spears (15-38); Christina Aguilera (44-51,80); Destiny's Child (52-62); Janet Jackson (67-73); Jennifer Lopez (74-78,82); Madonna (84-86); and others	(10,14) (16,17) (74,77) (75,82) (88,89) (93,94)	(37) is from same recording as (15,21,22) but with sound effects
Reggae	(03,54) (05,56) (08,57) (10,60) (13,58) (41,69) (73,74) (80,81,82) (75,91,92)	(07,59) (33,44) (85,96)	Bob Marley (00-27,54-60); Dennis Brown (46-48,64-68,71); Prince Buster (85,94-99); Burning Spear (33,42,44,50, 63); Gregory Isaacs (70,76-78); and others	(23,55)	last 25 s of (86) are useless
Rock	(16) is Metal (58)		Morphine (0-9); Ani DiFranco (10-15); Queen (16-26); The Rolling Stones (28-31,33,35,37); Led Zeppelin (32,39-48); Simple Minds (49-56); Sting (57-63); Jethro Tull (64-70); Simply Red (71-78); Survivor (79-85); The Stone Roses (91-99)		jitter (27)

Figure 15: Repetitions and distortions in GTZAN. Samples numbers are in parentheses.

5.1.3 Distortions

Distortions were found listening to every samples of the GTZAN dataset. However, the author highlights that one specific sample (*Reggae 86*) is so severely distorted that its last 25 seconds are useless.

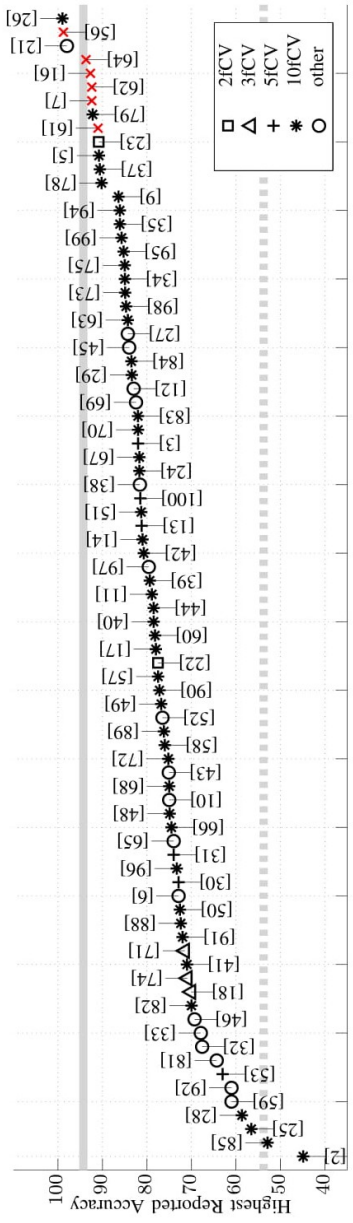


Figure 16: Highest classification accuracies.

5.2 Benchmark

We decide again to refer to the *Sturm*'s research [1] for our benchmark. As said before, the **GTZAN dataset** is one of the most used the in Music Genre Recognition contest: however, the author noted that, among 100 works, only five of them indicated that someone has listened to at least some samples of the GTZAN; also, very few works mention specific faults in the dataset.

5.2.1 Reviewing Evaluation

In the 100 works selected, 96 employ the **Classification** task: most measure performance is classification **accuracy** computed from **k-fold cross-validation** (kfCV). Other works measure classification accuracy using split of data like train and test or train/validation/test sets.

In figure 16 the highest classification accuracies of the 100 works are showed. However, Sturm find out that half of these didn't use other dataset, which means that a majority of evaluations can provide no conclusions about the performance of a system on other datasets.

Since few of these works consider the faults of the GTZAN dataset, it is not difficult to see that those evaluation metrics are not reliable. For instance, when **replicas** are distributed across train and test sets, the evaluation of some systems can be more biased than others e.g. *nearest neighbor* classifier will see zero distance in the training set. Also if replicas are present in the test set only, this will affect the evaluation: if a sample it is classified in/correctly then also the replica will be classified in/correctly.

5.2.2 Method

The author use three classifiers with the same features: **nearest neighbor** (NN), **minimum distance** (MD) and **minimum Mahalanobis distance** (MMD). The features are 13 MFCCs, the same used in our approach. Also, the evaluation considers two state-of-the-art MGR systems that produce highest classification accuracies using GTZAN:

- Maximum a posteriori classification of scattering coefficients (MAP-sCAT) which uses a Bayesian framework.
- Sparse representation classification with auditory temporal modulations (SRCAM).

Sturm evaluates each system using GTZAN with two different kinds of partitioning: ten realization of 2-fold cross-validation (**ST**) and ST without the distortions and the exact and recording repetitions (**ST'**). First, systems are evaluated by **normalized accuracy**, since not classes are equally represented in the test sets.

In figure 17 it is immediately clear that estimates of the classification accuracy for each system are affected by the fault of GTZAN. Since the SRCAM model is the one that seems to perform better, the author decides to compute other figure of merits (FoG).

Figure 18 shows how we finally retrieve our benchmark.

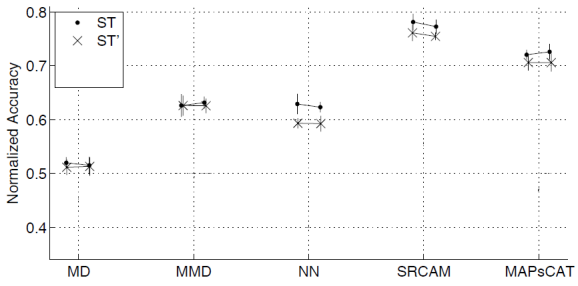


Figure 17: Normalized accuracy of each system for each fold (left and right) of different partitioning (legend).

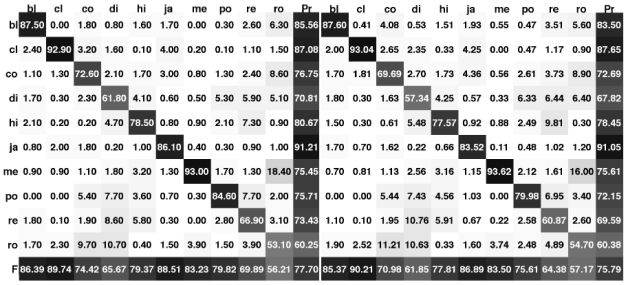


Figure 18: Precision (Pr), F-score (F) and normalized accuracy (bottom right corner) for SRCAM, evaluated with ST (table on the left) and without repetition ST' (table on the right).

5.3 Our Dataset

In addition to testing the effectiveness of our models on the famous GTZAN dataset, for educational purposes we tried ourselves to build a musical dataset. The first and fundamental step to achieving this goal was clearly to collect raw data. The idea is this: for each genre we considered a YouTube playlist of copyright-free songs, and we collected the links of all the playlists in a specially formatted text file. The code then, for each playlist, and therefore for each genre, proceeded to download the video files. The latter are then converted into audio format, mono channel and 22050Mhz sampling, just like the GTZAN dataset. Finally, the entire resulting track is cut into a segment with a fixed duration of 30 seconds.

Figure 19: Workflow to retrieve our dataset.

The size of playlists, and therefore of the classes of future models, are initially really unbalanced; for this reason we proceeded to eliminate the 2 smaller classes, leaving a total of 10 classes, just like the GTZAN dataset. Clearly, some classes are in common with respect to the other dataset, while others are different, so it is not possible to make an absolute comparison between the two datasets as some genres are more distinguishable than others. Finally, for too large classes, up to 100 audio tracks were undersampled, and a balanced loss with respect to the size of the classes was used in the models. So here's the before and after:

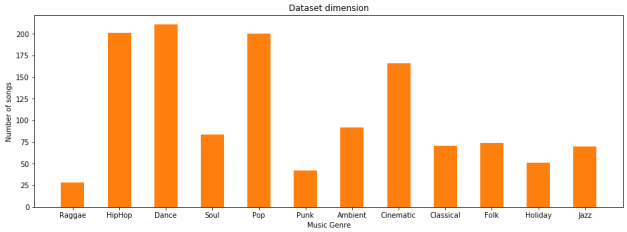


Figure 20: Before undersampling.

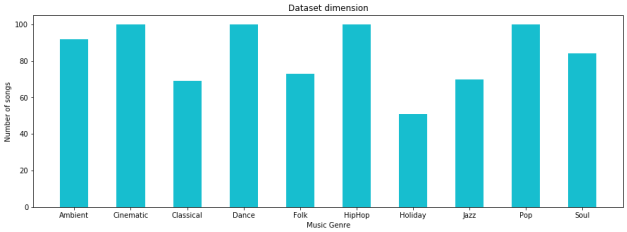


Figure 21: After undersampling.

Here are some interesting statistics on the length of the tracks before the cut to 30 seconds, specifying that our dataset does not contain more tracks for every single song:

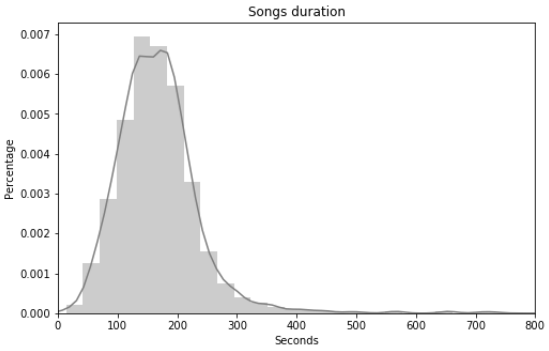


Figure 22: Distribution of the tracks' duration.

6 Experimental Results

We decide to train and test each network using different procedures:

- **K-fold cross-validation:** as done in the benchmark with $k = 2$.
- **Train/Validation and test split:** we used 20% of the dataset as *validation set*, 25% of the dataset as *test set* and the remaining samples of the dataset as *training set*.

For each method several *figures of merit* are computed: *accuracy, recall, precision* and *F1-score*. During the training the **early stopping** procedure is applied: we set a large number of epochs and we stop when the *validation loss* (loss on the training set in the k-fold case) doesn't increase for 5 consecutive epochs.

6.1 Multi-layer Perceptron Network

The settings described in section **4.1** dramatically improve the accuracy and further reduce over-fitting (**Figure 23 -24 25**).

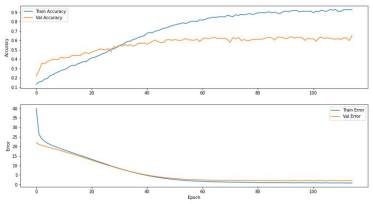


Figure 23: First dense training

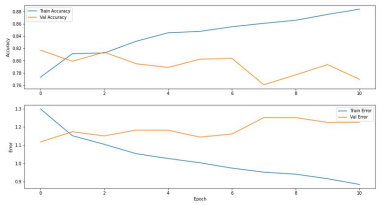


Figure 24: Sparse training

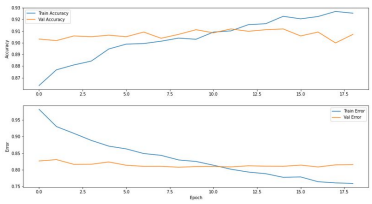


Figure 25: Last dense training

Figure 26: Train/validation accuracy and loss over epochs

We decide to test the network of 25% of the dataset. The MLP reaches an accuracy of approximately 92% on the test set (of the full GTZAN dataset). Looking at different metrics, it is possible to gain further insights into the problem.

As it's possible to see in the confusion matrix (**Figure 27**) the most mislabelled classes are country, jazz and rock.

Looking at the classification report (**Figure 28**), it is possible to see that the classes with the lowest f1 score are class 2 (country), class 5 (jazz) and class 9 (rock).

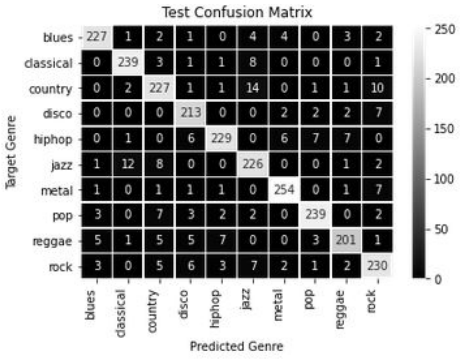


Figure 27: Confusion Matrix of all classes

```
Accuracy on test set is: 0.9171005487442017
Loss on test set is: 0.7882264256477356
              precision    recall  f1-score   support

           0       0.92      0.89      0.91       235
           1       0.96      0.94      0.95       256
           2       0.88      0.85      0.86       244
           3       0.94      0.92      0.93       259
           4       0.90      0.94      0.92       236
           5       0.87      0.92      0.89       259
           6       0.96      0.94      0.95       251
           7       0.93      0.94      0.94       259
           8       0.93      0.92      0.92       254
           9       0.87      0.91      0.89       244

    accuracy                           0.92      2497
   macro avg       0.92      0.92      0.92      2497
weighted avg       0.92      0.92      0.92      2497
```

Figure 28: Classification report.

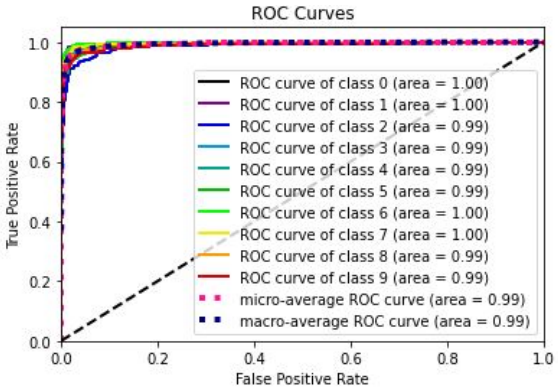


Figure 29: ROC curves: one curve for each class.

6.2 Convolutional Neural Network

The following results are obtained by the CNN model on the GTZAN Dataset.

The 20% of the dataset is used as validation set. The percentage of samples used as training set instead is 55%. As shown in the plots, in the train/validation scenario, during the training phase, the accuracy increases over time while the loss decreases.

```
loss: 0.4210 - accuracy: 0.8568 - val_loss: 0.8941 - val_accuracy: 0.7063
```

Figure 30: Train/Validation accuracy and loss results

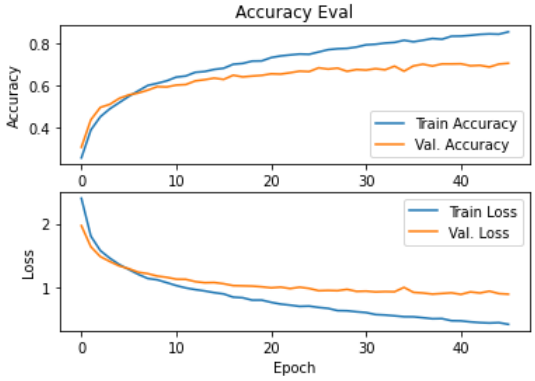


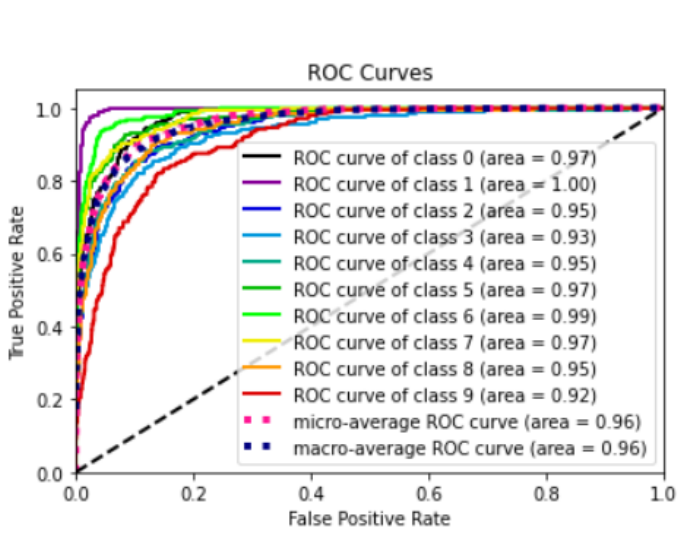
Figure 31: Train/validation accuracy and loss over epochs

```
Accuracy on test set is: 0.7288746237754822
Loss on test set is: 0.8087599277496338
              precision    recall  f1-score   support

           0       0.76      0.78      0.77       229
           1       0.88      0.92      0.90       254
           2       0.67      0.61      0.64       273
           3       0.69      0.61      0.64       252
           4       0.70      0.69      0.70       244
           5       0.80      0.80      0.80       256
           6       0.85      0.81      0.83       242
           7       0.77      0.76      0.76       268
           8       0.63      0.71      0.67       243
           9       0.55      0.61      0.58       236

    accuracy                           0.73      2497
   macro avg       0.73      0.73      0.73      2497
weighted avg       0.73      0.73      0.73      2497
```

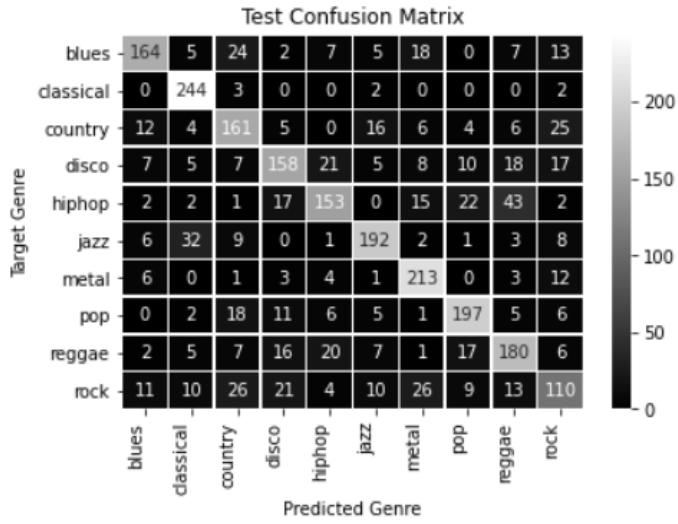
Figure 32: Perfomances of CNN on test set

After this step, we can evaluate our model on the test set with some metrics such as **accuracy**, **precision**, **recall** and **f1-score**. The 20% of the original dataset is been used as test set.

The results are in line with our benchmark. Finally we show the **ROC curve** and the **confusion matrix** in order to highlight the result of each class.



(a)



(b)

Figure 33: (a) Roc curve and (b) Confusion matrix

We can assert that results are in line with other models proposed. Rock is the most missclassified genre, while classical music, thanks to its distinctive patterns, obtains best performances.

6.3 Recurrent Neural Network

Now we show the different results obtained by the RNN with the full GTZAN dataset.

```
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 1.1216895580291748 - Accuracy: 71.962618827819825%
------------------------------------------------------------------------
> Fold 2 - Loss: 1.0719726085662842 - Accuracy: 70.51281929016113%
------------------------------------------------------------------------
Average scores for all folds:
> Accuracy: 71.23771905899048 (+- 0.7248997688293457)
> Loss: 1.0968310832977295
```

Figure 34: K-fold cross-validation (with k = 2) metrics.

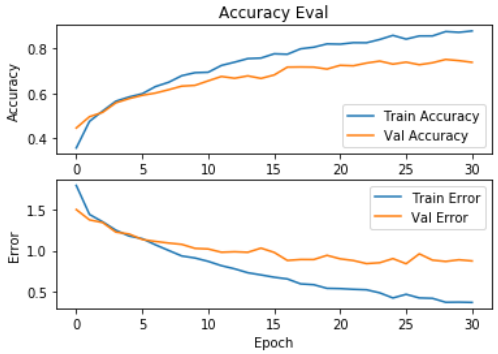


Figure 35: Train/validation accuracy and loss over epochs.

Figure 34 shows the results obtained using k-fold cross validation, with 2 folds, while figure 35 shows the training phase of the RNN using 20% of the dataset as validation set (orange lines) and 65% of the dataset as training set (blue lines). It is clear how the accuracy increases over time, while the loss decreases, which is what we are expecting.

The results show good accuracy in both cases, so we decided to test it on the remaining 25% of the dataset.

```
Accuracy on test set is: 0.751702070236206
Loss on test set is: 0.8410719633102417
              precision    recall  f1-score   support

           0       0.76      0.83      0.79       239
           1       0.83      0.95      0.88       241
           2       0.59      0.67      0.62       259
           3       0.78      0.71      0.74       263
           4       0.65      0.80      0.72       222
           5       0.79      0.66      0.72       266
           6       0.87      0.83      0.85       241
           7       0.87      0.85      0.86       249
           8       0.79      0.76      0.77       278
           9       0.61      0.49      0.55       239

    accuracy                           0.75      2497
   macro avg       0.75      0.75      0.75      2497
weighted avg       0.75      0.75      0.75      2497
```

Figure 36: Classification report.

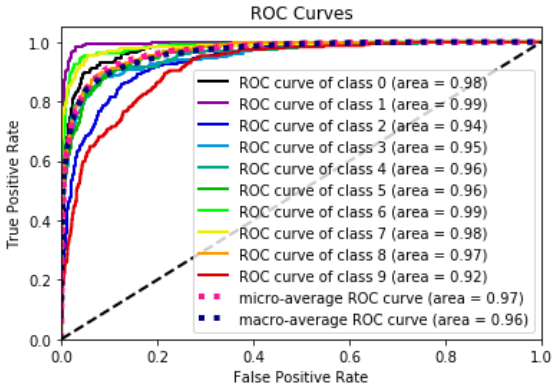


Figure 37: ROC curves: one curve for each class.

Figure 36 shows the classification report resulting from the testing from which we can read the overall accuracy, precision, recall and F1-score. They are pretty good performances if we do a comparison with our benchmark. Let's see the ROC curve: figure 37 confirms that the model has difficulty in classifying the Rock class, this is a common problem. Instead, the RNN has very high performance in classifying the class 1, which corresponds to the Classical musical genre, and this is visible also in the classification report. Now let's better understand the misclassification, plotting the confusion matrix of all classes:

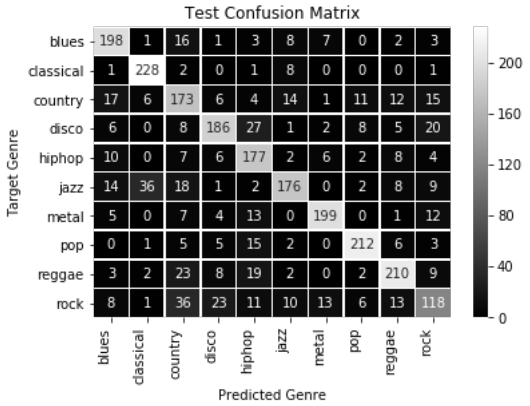


Figure 38: Confusion matrix of all classes.

6.4 Convolutional Recurrent Neural Network

In this section we show the different results obtained by the **CRNN**.

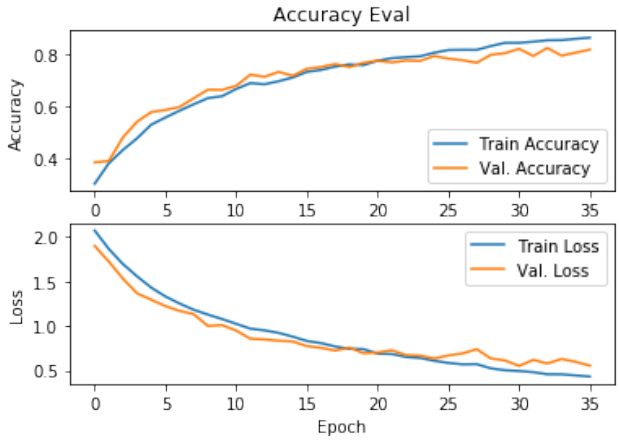


Figure 39: Train/validation **accuracy** and **loss** over epochs.

Figure 39 shows the training phase of the CRNN using 20% of the dataset as validation set (orange lines) and 65% of the dataset as training set (blue lines).

It is clear how the accuracy increases over time, while the loss decreases, which is what we are expecting.

Setting 500 epochs, after 163 of them:

- **Training:** 0.4321 of **loss** and 0.8625 of **accuracy**.
- **Validation:** 0.5552 of **loss** and 0.8171 of **accuracy**.

The results show good accuracy in both cases so the model can be considered reliable. We decided to test it on the remaining 25% of the dataset.

Figure 40 shows the classification report resulting from the testing from which we can read the overall *accuracy*, *precision*, *recall* and *F1-score*. They are pretty good performances if we do a comparison with our benchmark. Analyzing the classification report we can see very low performances in classifying the class 9, which correspond to the *Rock* class. Let's see the **ROC curve**.

```
Accuracy on test set is: 0.8229875564575195
Loss on test set is: 0.5482540718098139
              precision    recall  f1-score   support

           0       0.87      0.81      0.84       231
           1       0.93      0.96      0.94       247
           2       0.70      0.87      0.77       260
           3       0.81      0.78      0.80       262
           4       0.86      0.79      0.82       272
           5       0.90      0.86      0.88       231
           6       0.87      0.89      0.88       238
           7       0.88      0.81      0.84       254
           8       0.75      0.86      0.80       266
           9       0.71      0.58      0.64       236

    accuracy                           0.82      2497
   macro avg       0.83      0.82      0.82      2497
weighted avg       0.83      0.82      0.82      2497
```

Figure 40: Classification report.

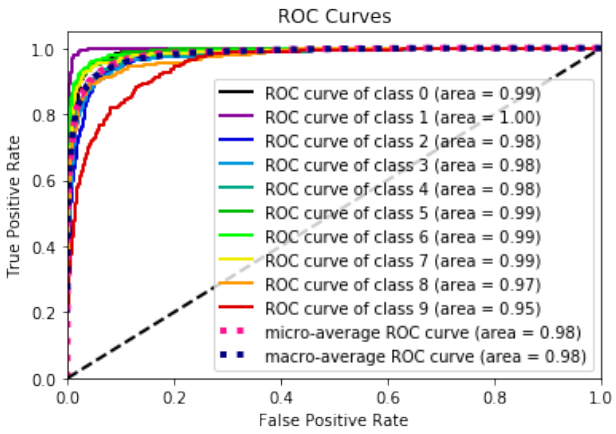


Figure 41: ROC curves: one curve for each class.

Figure 41 confirms that the model has issues in classifying the Rock class. Instead, the CRNN has very high performance in classifying the class 1, which corresponds to the *Classical* musical genre, and this is visible also in the classification report.

Now let's better understand the misclassification, plotting the **confusion matrix** of all classes. In figure 42 it is possible to see that some of the rock tracks are misclassified as metal: in some sense the metal is a sub-genre of rock, and this is a possible explanation of the misclassification. A lot of rock songs are also misclassified as *country* tracks.

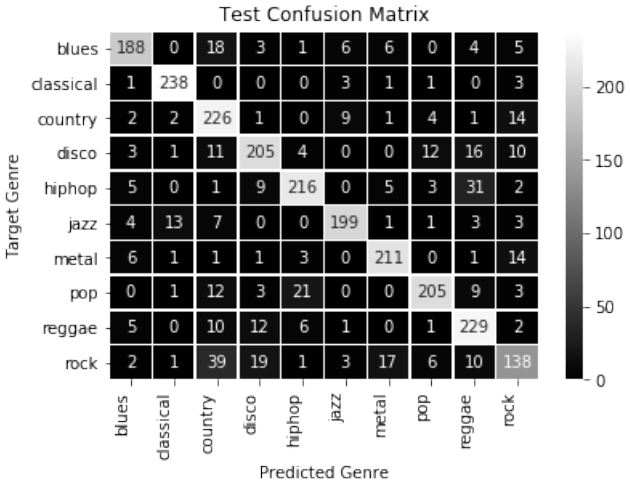


Figure 42: Confusion matrix of all classes.

7 Results on other datasets

Here we present a brief report of the accuracy reached on our dataset and on the clean GTZAN dataset

	GTZAN	Clean GTZAN	Our dataset
MLP	0.92	0.95	0.86
CNN	0.73	0.72	0.38
RNN	0.75	0.74	0.45
CRNN	0.82	0.72	0.53

Figure 43: Accuracy of different models on different datasets

8 Conclusions and Future Work

We have observed that the most mislabelled class is rock. We suppose this happens because a plethora pieces of music from various different genres are labelled as rock music: as an example consider the difference between The Beatles and Led Zeppelin, which are both classified as rock music. Moreover there are many different rock subgenres like rock-metal, so misclassification happens easily. On the other hand all models have a good performance in classifying the classical genre. We suppose this is due to the distinctive patterns of classical music, which seldom overlap with other genres. Lastly, we observe that the best performance is achieved by the multi layer perceptron trained with the dense-sparse-dense technique.
Secondly, we have noticed that having a clean dataset improves performances for some models and worsen it for other model. This is due to the fact that the clean GTZAN dataset has less datapoints and classes are slightly unbalanced.
On the other hand, performances on our dataset are overall worse because the genres are not the same as GTZAN dataset. Our genres are Ambient, cinematic, classical, dance, folk, hip-hop, jazz, pop, soul, holiday. As it is possible to notice, the genres in our dataset are not as good defined as the GTZAN genres. This choice was made to test our models also on real data, and not only on benchmark data. As an example, the genre classical and cinematic overlaps, since a lot of music used in movies is classical music. The same can be said about the Holiday genre: this class boundaries are not strongly defined, and the genre overlaps with many different genres.

As for future work, we think that it is useful to develop a model which can perform multi-labelling classification, since a song may belong to multiple genres. Moreover we want to further investigate the usage of recurrent neural networks in our work. NLP state-of-the-art models learn sentences in bidirectional way. This could be tried with music by training the model both on the standard tracks and on the reversed tracks, thus providing more features.

The project work was split in the following way:

- Data Preprocessing and CRNN: Chiara Chiucchi;
- MLP: Emanuele Mercanti;
- CNN: Matteo Pascolini;
- RNN and custom dataset: Andrea Trianni.

9 References

1. The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use
2. Convolutional Recurrent Neural Networks for Music Classification
3. Music Artist Classification with Convolutional Recurrent Neural Networks
4. DSD: Dense-Sparse-Dense Training for Deep Neural Networks
5. Deep Learning for Audio