Università degli studi di Milano-Bicocca

Advanced Machine Learning

Final Project

Music Genre Classification using Convolutional and Recurrent Neural Networks

Authors:
Riccardo Ghilotti - 879259 - r.ghilotti@campus.unimib.it
Mattia Ingrassia - 879204 - m.ingrassia3@campus.unimib.it
Alessandro Isceri - 879309 - a.isceri@campus.unimib.it

January 18, 2026

Abstract

A challenging task in the music industry is genre classification; that is, detecting the music genre of an audio track. Thanks to new technologies developed in the Deep Learning Era, it is possible to create a model that automatically retrieves the correct genre from raw data. To address this problem, the proposed approach focused on several Deep Learning methods, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). This study utilized the GTZAN dataset, which is composed of several audio tracks that have been assigned different representations, such as MEL spectrograms and tabular statistical features. To compare the obtained results, a baseline model for both CNNs and RNNs was developed. The proposed approach explored different techniques, namely data augmentation, noise injection, and data normalization, to improve the models' performance. Finally, to enhance robustness, generalization, and overall performance, three ensemble methods have been designed. As expected, the proposed models outperformed the baseline ones by a noticeable percentage.

1 Introduction

Music experts have been trying for a long time to understand sound and what differentiates one song from another. The main objective of the project is to build a system that classifies a song based on its genre. There could be several scenarios where this system might be useful; for example, labeling new songs automatically, creating custom playlists based on genres, giving customers a clear idea of their music tastes, and some existing recommendation systems could leverage a better classification technique to retrieve similar songs. There are already some systems that mimic or implement some of these processes in modern applications, but they usually perform poorly.

To achieve correct genre classification, two approaches have been explored based on different representations of the same data: the first used MEL spectrograms of the audio tracks, the second leverages the intrinsic sequentiality of the data by using features extracted from different time steps of the tracks.

To expand the number of instances, some data augmentation techniques were applied for both previously mentioned datasets; this should result in improved performance, robustness and generalization.

Finally, leveraging the previously trained models, three ensemble methods were created for similar reasons.

The document is structured in different sections:

- In section 2, different datasets, augmentation techniques, and data pre-processing are explored in detail.
- In section 3, the motivations behind the proposed solutions are presented.
- In section 4, performance of developed models are reported and visualized using radar graphs, histograms and tables.

- In section 5, the results are analyzed by comparing the performance of different models. The discussion highlights how variations in data augmentation techniques and model architectures contribute to the observed differences in performance.

2 Datasets

The dataset used for this project is the GTZAN dataset [1], one of the most widely used public datasets for evaluation in machine listening research for music genre recognition (MGR). The files were collected in 2000-2001 from a variety of sources, including personal CDs, radio, microphone recordings, in order to represent a variety of recording conditions.

The dataset holds the `.wav` files representing the recorded songs in the `genres_original` folder, and a folder `images_original`, containing the MEL spectrograms that were used to train the CNN model. The dataset also includes two `.csv` files containing features extracted directly from the `.wav` file of each song. In particular, one file includes features computed on the whole song (30 sec), while the other one contains the same features computed on 10 splits of the original song (each 3 seconds long).

The `.csv` contains 60 columns, where 57 were features computed from the data:

- **Column 0 - Filename**: Reference to the original `.wav` audio file.
- **Column 1 - Length**: Length of the sequence of vibrations in varying pressure strengths.
- **Columns 2-3 - Chroma STFT (mean, variance)**: Statistics of the Chroma Short-Time Fourier Transform.
- **Columns 4-5 - RMS Energy (mean, variance)**: Root Mean Square (RMS) energy, indicating the average loudness of the audio signal.
- **Columns 6-7 - Spectral Centroid (mean, variance)**: Describes the center of mass of the spectrum and is often associated with the perceived brightness of the sound.
- **Columns 8-9 - Spectral Bandwidth (mean, variance)**: Measures the spread of the spectrum around its centroid, indicating how broad the frequency content is.
- **Columns 10-11 - Spectral Roll-off (mean, variance)**: Frequency below which a specified percentage of the total spectral energy is contained; it describes the shape of the spectrum.
- **Columns 12-13 - Zero-Crossing Rate (mean, variance)**: Rate at which the signal changes sign, from positive to negative or vice versa.
- **Columns 14-15 - Harmonic Components (mean, variance)**: Represents harmonic content related to timbre (sound color), capturing tonal characteristics beyond human pitch perception.
- **Columns 16-17 - Perceptual Features (mean, variance)**: Features related to rhythmic and emotional aspects of the audio signal.

- **Column 18 - Tempo**: Estimated tempo (in beats per minute), extracted using a dynamic programming beat-tracking algorithm.
- **Columns 19-58 - MFCCs (mean, variance)**: Mel-Frequency Cepstral Coefficients (20 coefficients, each with mean and variance), describing the overall shape of the spectral envelope and modeling perceptual characteristics of human hearing.
- **Column 59 - Label**: Musical genre, one of: Blues, Classical, Country, Disco, Hiphop, Jazz, Metal, Pop, Reggae, Rock.

2.1 MEL Spectrograms

In order to improve the quality and numerosity of the dataset, data cleaning and two augmentation pipelines were applied.
First of all, a corrupted instance (jazz-54) was excluded from the dataset. Since the images were padded with a white border, the cleaning pipeline removed it. Additionally, the channels were scaled to obtain values in the range [0, 1]. Regarding the augmentation pipelines, they were applied to both the original and cropped images. The first one was composed of 3 transformations:

- **Frequency masking**: masks random segments along the frequency axis of a spectrogram; this transformation has a probability of 60% of being applied.
- **Time masking**: masks random segments along the time axis of a spectrogram; this transformation has a probability of 60% of being applied.
- **Time Reverse**: reverses the time axis of a spectrogram image; this transformation has a probability of 50% of being applied.

And the second one always applied one of these three transformations:

- **Salt & Pepper**: replaces random pixels with either 1 or 0.
- **Gaussian**: adds gaussian-distributed noise.
- **Poisson**: adds poisson-distributed noise generated from the data.

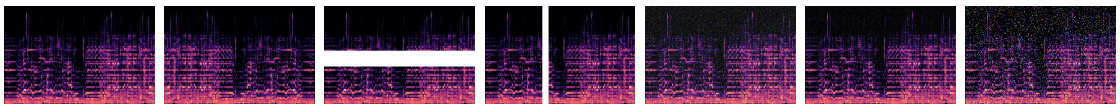

Figure 1: Data augmentation techniques applied in the pipeline

In image 1, an example image (on the left) and the resulting images of the various transformations applied to the original image are shown.

A total of five datasets were obtained:

- **Original**: original images contained in the GTZAN dataset.

- **Original Augmented**: augmented images obtained starting from the original images.
- **Cropped**: cropped version of the original images.
- **Cropped Augmented**: augmented images obtained starting from the cropped version of the original images.
- **Cropped Augmented & Noise Injection**: noisy and augmented images obtained starting from the cropped version of the original images.

The resulting datasets are available at [2].

2.2 Tabular sequential data

Regarding the tabular data, the file `features_3_sec.csv`, which contains features for 10 time steps of each track was used; in particular, it contains 10 rows per instance, each corresponding to a time step and is composed of 57 features; it was necessary to aggregate them into a 3D tensor of shape (n_instances $\times$ 10 time steps $\times$ 57 features) to be used as input for the RNNs. Again, the corrupted instance (jazz-54) was excluded from the dataset.

After reshaping the dataset, the data were normalized to ease the convergence of the learning algorithm; a data scaler was fitted on the flattened version of the training data and used to scale both the test and validation data.

Additionally, an augmented version of the dataset was produced; the augmentation was done by firstly computing the standard deviation of each feature at each time step on the normalized training data. Secondly, the original features were augmented by adding a random value sampled from an interval defined using the corresponding standard deviation.

A total of two transformations were applied to the tabular dataset, obtaining:

- **Original**: original instances contained in the `features_3_sec.csv` file.
- **Scaled**: normalized instances contained in the `features_3_sec.csv` file.
- **Scaled & Noise Injection**: noisy and normalized images contained in the `features_3_sec.csv` file.

3 The Methodological Approach

To tackle the problem of audio track classification, given the datasets of images and sequential data, different models were tested.
Image classification was performed by two different CNN architectures, and the sequential data was fed to a pair of RNN architectures to leverage the respective data structures.
To ease the image classification task and boost the performance of the CNN models, the data cleaning and augmentation pipelines described in section 2 were applied. A similar approach, also described in section 2, was used to look for improvements in the RNNs performances. Finally, three ensemble methods were defined, starting from the best performing models.

3.1 CNNs

The naive and optimized architectures are respectively reported in figures 2 and 3; the naive architecture was applied to the original images to obtain a baseline model (Model 0.0) to compare the performances of other implemented models.
Instead, the optimized architecture was trained on all datasets described in section 2.1 to evaluate the effectiveness of the presented transformations (Model 1.0, 1.1, 1.2, 1.3 and 1.4).

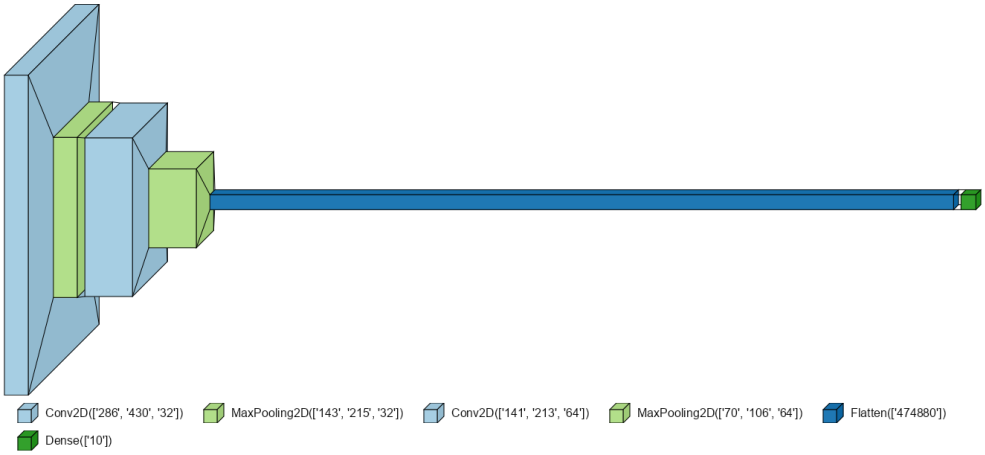

Figure 2: Structure of the naive CNN

The best performing model (Model 1.4) utilizes the optimized architecture trained on the **Original Augmented** dataset. In particular, this model was used to create a hybrid (Model 2.0), which leverages the features extracted from the former one and the handcrafted features fetched from the `features_30_sec.csv` file.

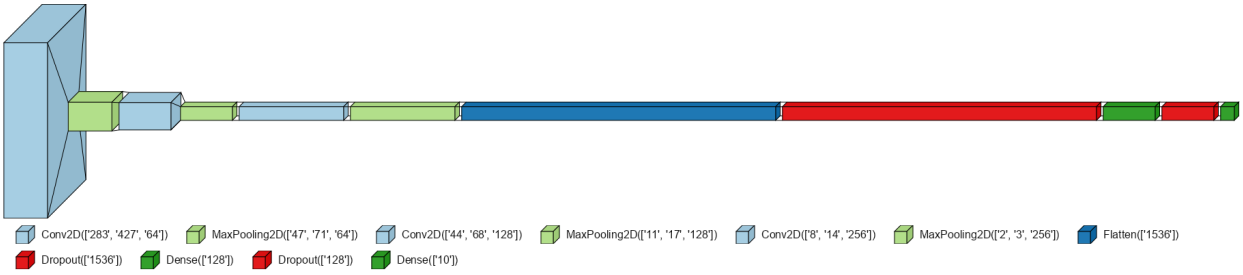

Figure 3: Structure of the optimized CNN

The latter model was used in the ensemble methods, and its architecture is reported in figure 4.

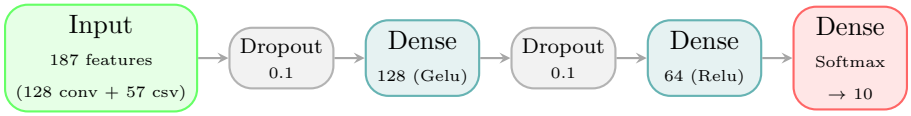

Figure 4: Structure of the HandCrafted model

3.2 RNNs

The naive and optimized architectures are respectively reported in figures 5 and 6; the naive architecture was applied to the original dataset to obtain a baseline model (Model 0.0). Instead, the optimized architecture was trained on all datasets described in section 2.2 to evaluate the effectiveness of the presented transformations (Model 1.0, 1.1, 1.2).

Figure 5: Structure of the Naive RNN

The best performing model utilizes the optimized architecture trained on the **Scaled & Noise Injection** dataset. For this reason, the model was chosen to be part of the ensemble methods.

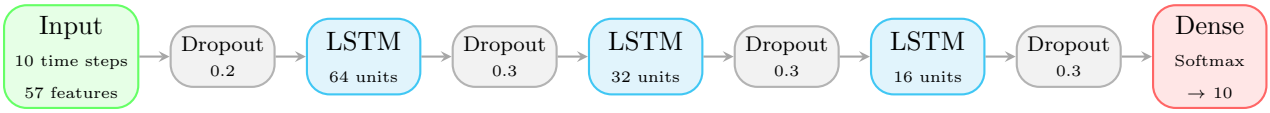

Figure 6: Structure of the optimal RNN

3.3 Ensembles

Since the best performing models differ in architecture, data structure, weight initialization, and other factors, they should make independent mistakes; hence, an ensemble model should perform better overall.

The three ensemble methods leverage slightly different strategies to combine the models:

- The first method combines the probability distributions given by the aforementioned models by summing them up and then selects the class corresponding to the peak value of that distribution (Model 3.1).
- The second method picks either the class that both models predict or, if they disagree, the most probable one out of both distributions (Model 3.2).
- The third method combines the probability distributions given by the aforementioned models by multiplying them and then picks the class corresponding to the peak value of that distribution (Model 3.3).

While testing the proposed solutions, several problems were encountered.

First, it was observed that the CNN models did not perform as expected on cropped images and achieved better results on the original dataset; this finding led to the augmentation of the original images. Secondly, since the task involves MEL spectrogram images, data augmentation was harder than expected. In particular, three types of data augmentation

were available, although these transformations worked well with the dataset, the number of possible options was fairly limited. Finally, since some tracks last only 29 seconds, certain instances in the `features_3_sec.csv` file consisted of nine temporal time steps instead of ten. Consequently, in the incomplete instances, the last temporal time step was inferred by computing the average of the previous time steps belonging to the same track.

4 Results and Evaluation

4.1 CNNs

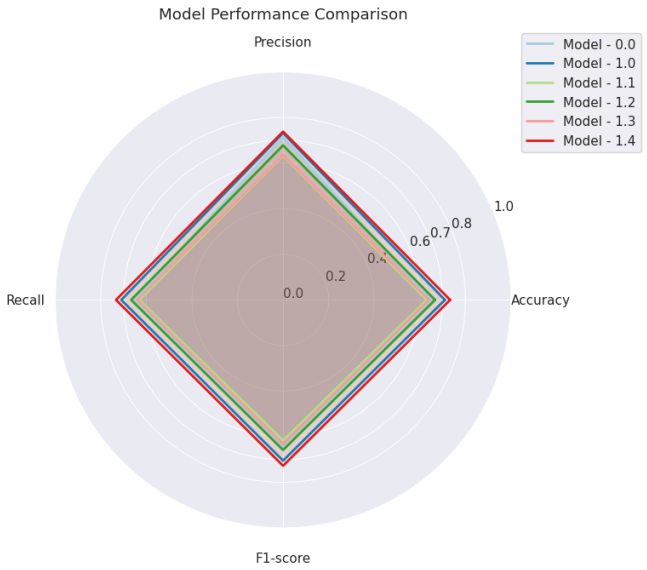

Figure 7: Comparison of the CNNs models

Table 1: Model performance (%)

Ver.	Acc.	Prec.	Rec.	F1
0.0	0.65	0.71	0.65	0.65
1.0	0.71	0.73	0.71	0.70
1.1	0.63	0.63	0.63	0.62
1.2	0.67	0.68	0.67	0.66
1.3	0.64	0.64	0.64	0.63
1.4	0.73	0.74	0.73	0.73

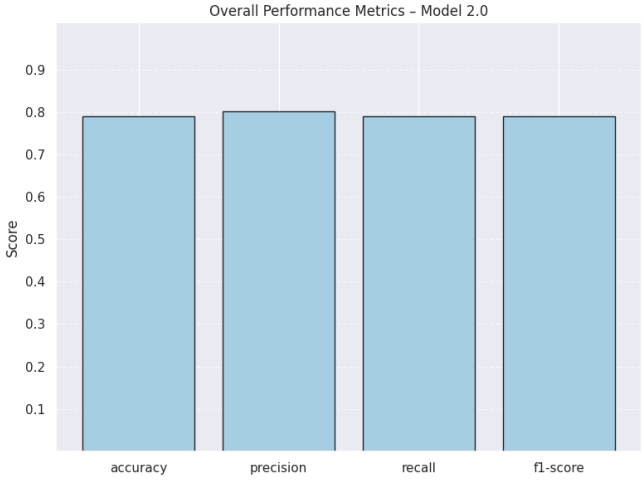

Figure 8: Performances metrics of the hybrid CNN

Table 2: Model performance (%)

Ver.	Acc.	Prec.	Rec.	F1
2.0	0.79	0.80	0.79	0.79

4.2 RNNs

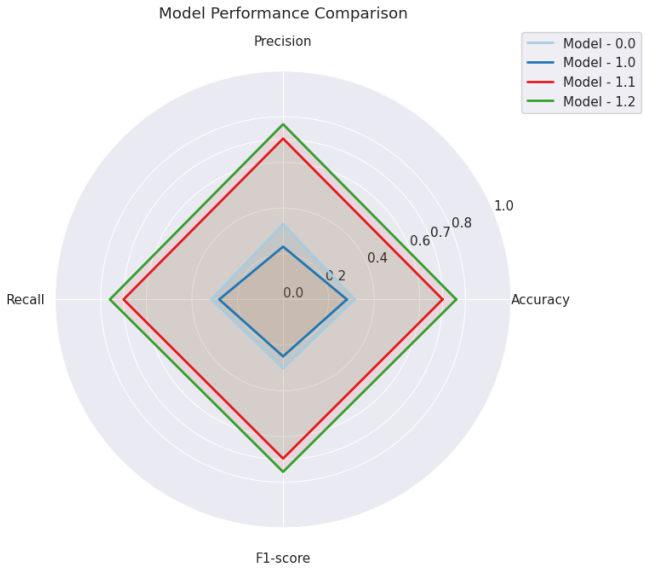

Figure 9: Comparison of the RNNs models

Table 3: Model performance (%)

Ver.	Acc.	Prec.	Rec.	F1
0.0	0.31	0.35	0.31	0.31
1.0	0.26	0.23	0.26	0.23
1.1	0.70	0.70	0.70	0.70
1.2	0.76	0.77	0.76	0.76

4.3 Ensembles

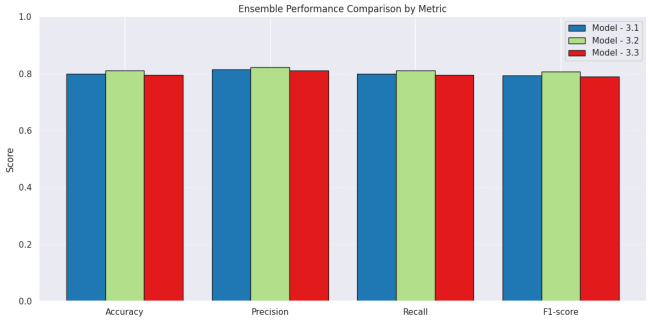

Figure 10: Comparison of the ensemble models

Table 4: Model performance (%)

Ver.	Acc.	Prec.	Rec.	F1
3.1	0.80	0.82	0.80	0.79
3.2	0.81	0.82	0.81	0.81
3.3	0.80	0.81	0.80	0.79

5 Discussion

5.1 CNNs

In figure 7 and in table 1, the results of the standard CNN models are reported. In particular, the baseline model (Model 0.0) achieves an accuracy and an F1-score of 65%. By simply changing the architecture (Model 1.0), a slightly better performance is observed; indeed, the accuracy increases by 6% and the F1-score by 5%.
Surprisingly, the same architecture applied to cropped images (Model 1.1) drops performance with respect to the baseline model, losing roughly 2% in all the computed metrics. However, the same architecture was trained on augmented cropped images (Model 1.2) to try and

boost the model's performance; although the general performance surpassed the baseline model and Model 1.1, the results are still disappointing.
Adding noise injection to the dataset (Model 1.3) did not help, as performance worsened compared to the baseline model. Since cropped images and noise injection brought a deterioration of the performances with respect to their counterparts, the decision to augment the original images directly, without applying noise injection, was made, leading to Model 1.4; this model performed better than all the previous ones, achieving 73% in accuracy and F1-score.
Lastly, the performances of the hybrid model (Model 2.0) are reported in figure 8 and in table 2; it can be observed that this last model towers over the others, with an accuracy and F1-score of 79%.

5.2 RNNs

In figure 9 and in table 3, the results of the various models are reported. In this case, the baseline model's performance is dire (Model 0.0), returning an accuracy and F1-score of 31%. In the learning curves, which can be found in the dedicated notebook (available at [3]), it can be seen that the network heavily overfits.
This leads to the next proposed model (Model 1.0) which leverages a more articulated architecture. The performances of this model are worse when compared to those obtained by the baseline model, but the learning curves are much better, with no overfitting in sight.
This suggested that perhaps normalizing the original dataset could improve the model performance significantly. This hypothesis was confirmed by Model 1.1, which outperformed both previous models and reached an accuracy and F1-score of 70%. Lastly, some noise was inserted into the normalized data to improve performance even further; as expected, the model trained on noisy data (Model 1.2) achieved another 6% over Model 1.1.

5.3 Ensembles

For the ensemble methods, the results can be found in figure 10 and in table 4. As theory indicates, the ensemble models are usually more robust and perform at least as well as the models that compose them. In this case, they performed even better, achieving a minimum of 80% in both accuracy and F1-score, with an 81% in the best case (Model 3.2).

6 Conclusions

In conclusion, through model architecture exploration, a well-performing model was found; meanwhile, although not perfect, the results were satisfying compared to the baseline models.

As many other works, this work has several limitations and possibilities for future improvement; first of all, it may be useful to expand the dataset to gather more tracks and genres, as this would ease the training phase and improve generalization.

Another possible improvement could be achieved by using other data augmentation and

noise injection techniques; hence the numerosity and diversity of the training dataset would raise and the models would benefit from this addition.

Thirdly, other ensemble techniques might be taken into consideration since the already obtained improvement in performance is not very relevant. For example, a weighted mean of the model's predictions could boost the obtained results.

Furthermore, it may be a good idea to try different approaches, such as audio or visual transformers, which could learn more in-depth relationships between data, or alternatively, to use a fine-tuned version of large pre-trained state-of-the-art CNN architectures that are commonly used for those types of tasks.

References

[1] A. Olteanu, "Gtzan dataset - music genre classification," Kaggle Dataset, 2020. [Online]. Available: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification

[2] R. Ghilotti, M. Ingrassia, and A. Isceri, "Gtzan 2.0," Kaggle Dataset, 2025. [Online]. Available: https://www.kaggle.com/datasets/mattiaingrassia/gtzan-2-0

[3] M. Ingrassia, R. Ghilotti, and A. Isceri, "Music genre classification," GitHub Repository, 2025. [Online]. Available: https://github.com/Mattia-Ingrassia/MusicGenreClassification