Comparative Analysis of CNN and RNN Architectures for Music Genre Classification

Seyedehsara Mirsepassi
Department of Computer Science, Università degli studi di Milano
seyedehsara.mirsepassi@studenti.unimi.it

Abstract

This study examines the effectiveness of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) units for music genre classification. It utilizes Mel-frequency cepstral coefficients (MFCCs) as input features and evaluates the impact of K-means clustering as an additional unsupervised classification method. The results show that CNNs outperform RNNs in both accuracy and training efficiency. Without K-means, CNN achieves 77.09% test accuracy with a training time of 43 milliseconds per step, while RNN reaches 69.52% test accuracy with 111 milliseconds per step. When K-means clustering is integrated, CNN accuracy decreases to 65.19%, while RNN accuracy significantly drops to 42.02%, indicating that clustering reduces performance in both models while maintaining CNN's advantage. These findings suggest that while K-means clustering is effective for grouping similar data points, it does not capture the complex spatial and temporal patterns necessary for accurate genre classification, ultimately limiting its effectiveness in deep learning-based approaches.

1 Introduction

Music genre classification is a fundamental task in audio pattern recognition with applications in music recommendation, automated tagging and digital library organization. The complexity of audio signals characterized by temporal variability and high dimensionality, makes this a challenging problem.

Traditional methods relied on manually extracted features such as pitch, rhythm and tone, coupled with classifiers like SVMs and decision trees. These approaches required domain expertise and struggled to capture complex patterns. Recent advances in deep learning, particularly CNNs and RNNs, have shown great results by learning patterns directly from raw data. CNNs excel at capturing spatial features from time-frequency representations, while RNNs, especially LSTMs, are effective at modeling temporal dependencies.

This study compares CNNs and RNNs on MFCC-based features, assessing accuracy, training time and complexity. Additionally, we explore K-means clustering as an unsupervised approach for genre classification, offering insights into feature grouping.

2 System Overview

2.1 Dataset description

Our study utilizes the GTZAN Genre Dataset, consisting of 1000 audio tracks, each 30 seconds in length. These tracks are categorized into 10 genres, with 100 tracks per genre. All tracks are 16-bit mono audio files with a sampling rate of 22050 Hz, stored in .wav format. The dataset was split into training, validation and test sets with a ratio of 60:15:25.

2.2 Data Preparation Pipeline

The data preparation process involved multiple stages to ensure the dataset was ready for model training and evaluation. Each audio file was processed as follows:

2.2.1 Segmentation:

Each 30-second track was divided into non-overlapping 3-second segments, creating 10,000 segments to enhance training diversity.

2.2.2 MFCC Extraction:

For each audio segment, 13 MFCCs were computed (sufficient to mimic how human ears perceive pitch) using the Librosa library to serve as input features for classification. This process was structured as follows:

- **Signal Resampling:** First, all audio signals were resampled to a standard rate of 22050Hz to ensure uniformity across tracks.
- **Framing and Windowing:** Then, each 3-second segment was divided into overlapping frames of 2048 samples with a hop length of 512 samples. A Hamming window was applied to each frame to smooth its edges to reduce spectral leakage at the frame edges. The framing and windowing process is automatically managed during the computation of MFCCs using librosa.feature.mfcc.
- **Spectral Transformation:** In the next section, the Short-Time Fourier Transform (STFT) was applied to each frame (Inside librosa.feature.mfcc) to turn the time-domain signal of each frame into a frequency-domain representation, which was then used to calculate the MFCCs for each frame.
- **Mel Scale Mapping and Logarithmic Transformation:** Afterward, Mel scale mapping and a logarithmic transformation were applied to the frequency-domain representation of each audio frame. The Mel scale is designed to align with human auditory perception, as we tend to notice smaller pitch changes more easily at lower frequencies, while changes at higher frequencies are less noticeable. This mapping adjusts the frequencies in the power spectrum to better match our perception of sound.

  In the librosa library, the librosa.feature.mfcc function automatically applies this Mel scale mapping, followed by a logarithmic transformation on the Mel-scaled energies to simulate the human ear's response to intensity variations.
- **Cepstral Coefficients:** Finally, the Discrete Cosine Transform (DCT) is applied to the Mel-scaled energies, compressing them into 13 coefficients per frame while preserving the most significant spectral features. This process is also incorporated within the librosa.feature.mfcc function, which automatically applies the DCT to the Mel-scaled frequencies. The result is stored in the "mfcc" variable, which contains the 13 coefficients per frame.

After extracting the MFCCs for all frames within all segments, the code flattens the MFCC matrices (from each segment) into a 1D array using .flatten(). This represents the entire segment's MFCCs as a single feature vector.

The dataset now contains 10,000 segments, each represented by MFCC sequences that capture both short-term spectral features and temporal dynamics, serving as input for the deep learning models.

2.2.3 K-means Clustering for Genre Classification:

K-means clustering provides an unsupervised method to analyze the grouping of MFCC-based features. This clustering process revealed patterns in how different genres relate to each other based on spectral characteristics.

For this task, K-means clustering groups segments into 10 clusters, each representing a genre. The results, along with the MFCC features and genre labels, are saved in a JSON file.

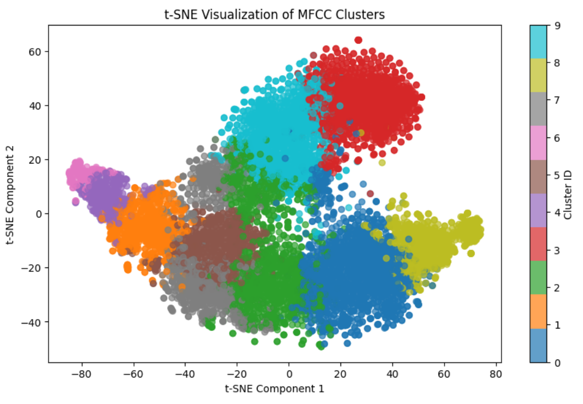


Figure 1: t-SNE Visualization of MFCC Clusters

In the analysis of cluster purity, classical music showed the highest purity, classical music showed the highest purity, particularly in Cluster 6 (92%) and Cluster 4 (70%), demonstrating clear genre separation. In contrast, genres such as metal (24% in Cluster 0, 51% in Cluster 8), reggae (22% in Cluster 2, 25% in Cluster 7), and hip-hop (23% in Cluster 9) exhibited greater overlap with other genres. This highlights the challenges of clustering music genres based on MFCC features, as some genres have distinct acoustic properties, while others share similarities that lead to blending across clusters.

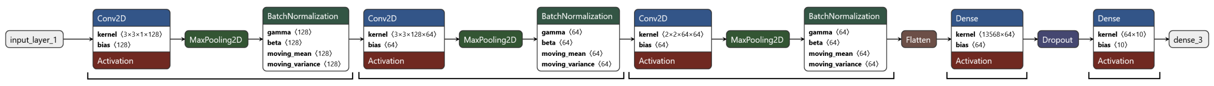


Figure 2: CNN Architecture

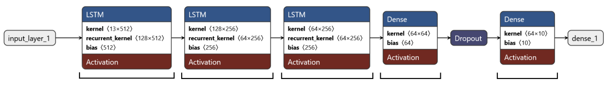


Figure 3: RNN (LSTM) Architecture

During data preprocessing, two types of JSON files were prepared: one with K-means clustering and one without, to evaluate the impact of clustering on the performance of CNN and RNN. In the version with K-means, each audio segment's MFCC features were clustered and accordinly the segment was assigned to a specific cluster. These cluster labels were then used alongside the original genre labels when splitting the dataset into training, validation, and test sets. The goal of clustering is to help the model find similarities in the data, which can improve its ability to recognize patterns shared by related genres.

2.3 Model Architectures

- **CNN Architecture:** The CNN model (Figure 2) consists of three convolutional layers with ReLU activation functions, each followed by max pooling and batch normalization.

  The first convolutional layer applies 128 filters (3×3) to extract low-level frequency-time patterns, followed by (3×3) max pooling with a stride of 2 to keep key features while reducing data size. Batch normalization stabilizes activations, improving training and reducing overfitting.

  The second convolutional layer uses 64 filters (3×3) to find more detailed features, while the third layer reduces the filter and pooling size to (2×2) for finer details. After convolution, a flatten layer converts feature maps into a 1D vector for fully connected layers. A dense layer with 64 neurons learns complex patterns, followed by a 20% dropout to prevent overfitting.

  The output layer has 10 neurons, each representing a music genre, with a softmax activation ensuring probability distribution across all genres.

  This architecture extracts meaningful features from the music, using convolutional layers for feature extraction, pooling layers for dimensionality reduction, and dense layers for classification. The inclusion of batch normalization helps stabilize training and accelerate convergence, while dropout reduces overfitting by forcing the model to rely on a broader set of features, ultimately ensuring good generalization to unseen data.

- **RNN (LSTM) Architecture:** The RNN model consists of three LSTM layers. The first LSTM layer, with 128 units, is set with return_sequences=True, meaning it outputs a sequence of hidden states at each step. This allows the model to capture long-term dependencies by retaining information from earlier steps.

  The second LSTM layer, with 64 units and also set with return_sequences=True, processes these sequences further.

  The third LSTM layer, with 64 units, has the default setting of return_sequences=False, converting the output into a fixed-length vector. This helps the model capture higher-level patterns and the overall structure of the input sequence.

  Following the LSTM layers, a fully connected dense layer with 64 neurons and ReLU activation, learns complex feature representations. To prevent overfitting, a dropout layer with a rate of 30% randomly drops connections during training.

  Finally, the output layer consists of 10 neurons with a softmax activation function, corresponding to the 10 music genres.

3 Performance Comparison of CNN and RNN Models with and without K-Means Clustering

The results show that CNNs outperform RNNs in both accuracy and training efficiency. Without K-means, CNN achieves 77.09% teast accuracy with a training time of 43 milliseconds per step, while RNN reaches 69.52% test accuracy with 111 milliseconds per step:

- **Without K-Means Clustering:**

Metric	Value
Validation Accuracy	92.74%
Test Accuracy	77.09%
Training Time	43 milliseconds per step
Trainable Parameters	154,186

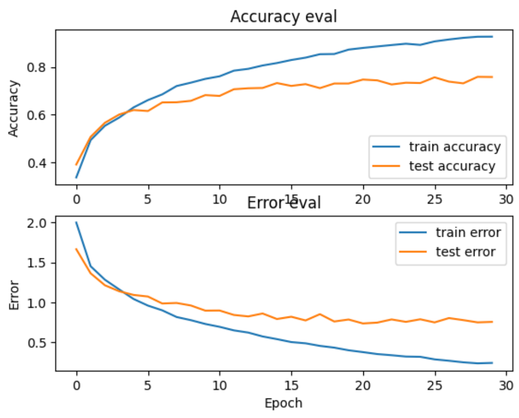


Figure 4: CNN Performance

Metric	Value
Validation Accuracy	82.54%
Test Accuracy	69.52%
Training Time	111 milliseconds per step
Trainable Parameters	159,946

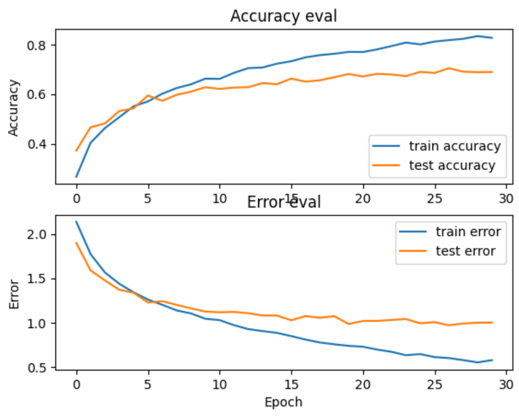
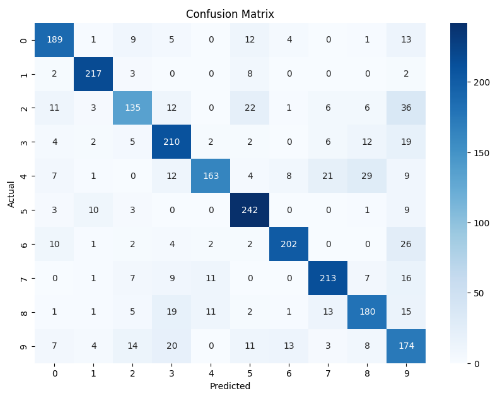


Figure 5: RNN Performance

The CNN confusion matrix shows high accuracy, with more correct predictions (diagonal) than errors. This highlights CNN's ability to capture spatial features in MFCC spectrograms, making it effective for analyzing frequency-time correlations.



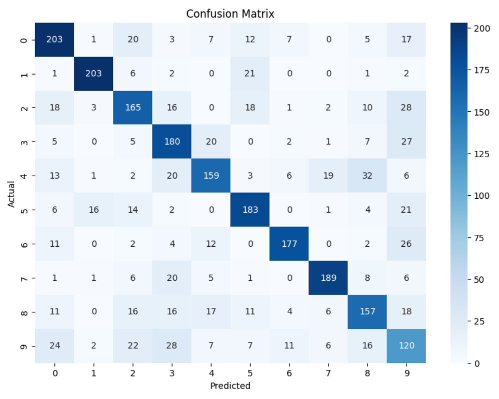
Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	189	1	9	5	0	12	4	0	1	13
1	2	217	3	0	0	8	0	0	0	2
2	11	3	135	12	0	22	1	6	6	36
3	4	2	5	210	2	2	0	6	12	19
4	7	1	0	12	163	4	8	21	29	9
5	3	10	3	0	0	242	0	0	1	9
6	10	1	2	4	2	2	202	0	0	26
7	0	1	7	9	11	0	0	213	7	16
8	1	1	5	19	11	2	1	13	180	15
9	7	4	14	20	0	11	13	3	8	174

Figure 6: CNN Confusion Matrix

Class	Samples	Precision	Recall
blues (0)	234	0.8077	0.8077
classical (1)	232	0.9004	0.9353
country (2)	232	0.7377	0.5819
disco (3)	262	0.7216	0.8015
hiphop (4)	254	0.8624	0.6417
jazz (5)	268	0.7934	0.9030
metal (6)	249	0.8821	0.8112
pop (7)	264	0.8130	0.8068
reggae (8)	248	0.7377	0.7258
rock (9)	254	0.5455	0.6850

Table 1: CNN Classification Report

In contrast, the confusion matrix for the RNN model shows moderate accuracy for some genres but higher misclassification rates compared to CNNs, particularly for genres with complex temporal structures like Jazz and Rock.



Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	203	1	20	3	7	12	7	0	5	17
1	1	203	6	2	0	21	0	0	1	2
2	18	3	165	16	0	18	1	2	10	28
3	5	0	5	180	20	0	2	1	7	27
4	13	1	2	20	159	3	6	19	32	6
5	6	16	14	2	0	183	0	1	4	21
6	11	0	2	4	12	0	177	0	2	26
7	1	1	6	20	5	1	0	189	8	6
8	11	0	16	16	17	11	4	6	157	18
9	24	2	22	28	7	7	11	6	16	120

Figure 7: RNN Confusion Matrix

Class	Samples	Precision	Recall
blues	275	0.6928	0.7382
classical	236	0.8943	0.8602
country	261	0.6395	0.6322
disco	247	0.6186	0.7287
hiphop	261	0.7004	0.6092
jazz	247	0.7148	0.7409
metal	234	0.8510	0.7564
pop	237	0.8438	0.7975
reggae	256	0.6488	0.6133
rock	243	0.4428	0.4938

Table 2: RNN (LSTM) Classification Report

The results show that RNNs struggle with MFCCs due to inefficient use of precomputed features and high computational demands. Despite more parameters, they don't outperform CNNs, making CNNs the better choice for MFCC-based music genre classification.

- **With K-Means Clustering:**

  When K-means clustering is integrated, CNN accuracy decreases to 65.19%, while RNN accuracy drops to 42.09%, indicating that clustering reduces performance in both models while maintaining CNN's advantage.

  This happens because K-means groups data by feature similarity without considering actual genre labels, which can add noise instead of improving structure. CNNs are better at recognizing spatial patterns in MFCCs, like harmonic structures, while RNNs (especially LSTMs) capture time-based dependencies important for genre classification. Since K-means treats features separately, it disrupts these spatial and temporal relationships, leading to weaker performance.

Metric	Value
Validation Accuracy	92.98%
Test Accuracy	65.19%
Test Training Time	131 milliseconds per step
Trainable Parameters	961,098

Table 3: CNN Performance (K-Means)

Metric	Value
Validation Accuracy	42.55%
Test Accuracy	42.09%
Training Time	1 second per step
Trainable Parameters	153,930

Table 4: RNN Performance (K-Means)

To improve K-means performance, data augmentation and Log-Mel feature extraction were also tested, but no improvement was observed. Data augmentation was applied to audio segments before extracting MFCCs, using time stretching, pitch shifting, and Gaussian noise to enhance variability and model generalization. However, it showed no significant gains for the CNN model and even reduced accuracy for the RNN model. Log-Mel spectrograms were also tested as an alternative to MFCCs, but their larger input size led to excessive memory usage and longer training times, particularly for the RNN model.

4 Conclusions

This paper presented a comparative analysis of CNN and RNN architectures with LSTM units for music genre classification using MFCCs as input data. The results demonstrate that CNNs outperform RNNs in both accuracy and training efficiency, achieving higher classification accuracy and significantly faster training times. This advantage stems from CNNs' ability to extract spatial features from MFCC spectrograms, which naturally align with their convolutional structure. Additionally, the findings reveal that K-means clustering negatively impacts both CNN and RNN performance, as it fails to capture spatial and temporal dependencies essential for effective classification. This study offers valuable insights into the suitability of different approaches for audio pattern recognition. Future research could explore hybrid models that integrate CNN and RNN components, investigate alternative clustering techniques, and leverage larger, more diverse datasets to further enhance model performance and generalization.

References

https://towardsdatascience.com/music-genre-recognition-using-convolutional-neural-networks-cnn-part-1-212c6b93da76

https://www.altexsoft.com/blog/audio-analysis/

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification