# Music Genre Classification: A Comparative Study between Deep-Learning and Traditional Machine Learning Approaches

Practical project for the Numerical Analysis for Machine Learning course (AY 23/24)

Gabriele Carrino, Giacomo Brunetta

## 1.Abstract

The goal of this project is the replication and validation of the results in the paper "*Music Genre Classification: A Comparative Study between Deep-Learning and Traditional Machine Learning Approaches*" by Dhevan S. Lau and Ritesh Ajoodha.
We ended up improving the results of the paper by introducing a normalization step before applying the techniques and by changing the underlying neural networks architectures.

## Contents

1.Abstract ........ 1
2.Work done in the paper ........ 2
3.Dataset ........ 2
4.Problems with the paper ........ 3
  4.1.Data preprocessing ........ 3
  4.2.Neural Network Architecture ........ 3
5.Methodology ........ 3
  5.2.Model testing ........ 3
  5.3.Machine learning techniques ........ 4
  5.4.Neural networks ........ 4
6.Results ........ 5
  6.1.Machine learning: 30s dataset ........ 5
  6.2.Machine learning: 3s dataset ........ 6
  6.3.Neural Networks ........ 7
7.Conclusion ........ 9
8.References ........ 10

## 2.Work done in the paper

The original research paper compared five off-the-shelf machine learning classifiers with two deep learning models on the task of music genre classification on the popular GTZAN dataset.
The **hypothesis** presented was that a **deep-learning approach would have outperformed the traditional models**, but the results obtained were lacking so the authors ended up neither rejecting nor accepting the hypothesis.

## 3.Dataset

For this project we used the same processed GTZAN dataset chosen in the original paper. [1]

The structure of the dataset is as follows: classification.

1. **genres original**: A collection of 10 genres with 100 audio files each, all having a length of 30 seconds.
2. **images original**: A collection of 1000 images representing the Mel Spectrograms of the audio files contained in genres original.
3. **2 CSV files –** Containing features of the audio files. One file has for each song (30 seconds long) 57 different features. The other file has the same structure, but the songs were split before into 3 seconds audio files (this way increasing 10 times the size of the dataset)

All audio files are classified in one of those 10 genres:

- Rock
- Classical
- Metal
- Disco
- Blues
- Reggae
- Country
- Hip-hop
- Jazz
- Pop

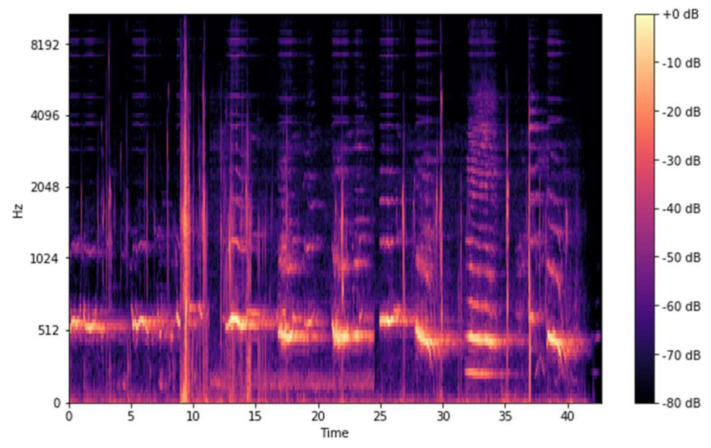


*Figure 1: Mel Spectrogram*

The dataset is perfectly balanced since there are 100 audio files per genre.

The features present in the dataset are typically the mean and the variance of metrics that can be obtained by applying signal processing techniques to the audio files.

The dataset, as in the original paper, was split into 80% training data and 20% test data. The training data was further split into 10-folds for cross-validation purposes.

## 4.Problems with the paper

We found two big classes of problems that led to the lacking results obtained by the authors, one related to data preprocessing and the other to the neural network architecture.

### 4.1. Data preprocessing

While trying to replicate the results we found that we could improve the performance (notably in the multilayer perceptron) by applying a **normalization** step to the dataset before feeding it to the models. From our testing we were able to deduce that the authors used *MinMaxScaler* (scales all the features between 0 and 1) instead of *StandardScaler* that instead normalizes (and centers) the data.

Another problem related to data preprocessing was the selection of a different subset of features for the machine learning and deep learning approaches, in particular the ML techniques were trained on all the 57 features available in the dataset, while the neural networks only on the **20 MFCCs**. This is a mistake because, by doing so, the authors are training the techniques on effectively different datasets and thus avoiding any reasonable comparison.

### 4.2. Neural Network Architecture

The authors built a single Convolutional Neural Network (**CNN**) and trained it on both the preprocessed features and the spectrogram images. We think that this is not the correct approach as CNNs are designed for **spatial data**, and forcing non-spatial data into this format might not accurately capture its underlying relationships. To solve this problem, we decided to use **two different architectures** for the two different datasets, a **classic Neural Network** with 4 hidden layers and a **CNN** like the one proposed in the paper.

The last problem was related to the CNN architecture, the authors fed the last convolutional block directly into the output of the model, this is not the best practice as the convolutional blocks perform feature extraction and the model is missing a layer that **aids classification**. We solved this problem by adding a **global average pooling** layer between the last block and the output and thus greatly improving the performance.

## 5.Methodology

### 5.2. Model testing

The dataset was split into **80%** training data and **20%** test data. As in the original paper, we reported the **cross-validation accuracy** (10 folds and 3 repetitions) and the **test accuracy** for each machine learning model. While we reported the **test loss** and the **test accuracy** of the deep learning models, along with the plots of the **training** and the **test loss** at each epoch. The loss function chosen for the deep learning models was the **Cross Entropy** function.

## 5.3. Machine learning techniques

This session provides a brief description of how the machine learning models have been implemented.

All the machine learning models are implemented using the **Scikit-Learn** Python library.

- The **Logistic Regression** (sklearn.linear_model.LogisticRegression) uses the **multinomial classification** strategy, which means that the model predicts the probability of each class directly. The solver is set to **LBFGS** (Limited-memory Broyden–Fletcher–Goldfarb–Shanno), which is an efficient Quasi Newton method for optimization. The loss function is regularized with **Ridge regularization** (l2 norm of the weights). The maximum number of iterations is set to 400.
- The **K-nearest Neighbor** (sklearn.neighbors.KNeighborsClassifier) algorithm is set to use only the nearest neighbor to the query point when making predictions (**1-Nearest Neighbor algorithm**).
- The **Support Vector Machine** (sklearn.svm .SVC) is set to use the **One-vs-One** strategy for multiclass classification.
- The **Random Forest** (sklearn.ensemble.RandomForestClassifier) model is composed **of 1000 decision trees** with a **maximum depth of 10**.
- The **Multilayer Perceptron** (sklearn.neural_network.MLPClassifier) is composed of an input layer of size 57, an hidden layer of size 5000, a second hidden layer of size 10 and an output layer of size 10. The solver is set to **LBFGS**, just like in the Logistic Regression and the activation function is **Relu**. The alpha parameter for the l2 regularization is set to 10e-5.

We have also tried other Neural Network architectures that perform better than the one proposed by Dhevan S. Lau et. Al. that performed better. The details are reported in the next section.

## 5.4. Neural networks

As said in section 4 we proposed two different architectures, one for the preprocessed features and one for the spectrogram images.

### 5.4.1. Deep Neural Network

The first model was a deep neural network with 57 input neurons, 4 hidden layers of (512, 256, 128, 64) and an output of 10 neurons. We applied a ReLU activation function and regularization dropout of 0.2 between each hidden layer and a dropout of 0.5 before the output one. To train the model we used the AdamW optimizer with a weight decay of 1e-5.

### 5.4.2. Convolutional neural network

The CNN base structure was exactly as in the paper i.e., an input layer followed by 5 convolutional blocks of:

- convolutional layer using a 3x3 filter, 1x1 stride and mirrored padding.
- ReLU activation function
- max pooling with a 2x2 windows size, 2x2 stride
- dropout regularization with a probability of 0.2

With filter sizes of (16, 32, 64, 128, 256) respectively.

The only difference was the addition of a global average pooling layer between the last block and the output with the scope of aiding classification.

**Global Average Pooling** is a pooling operation designed to replace fully connected layers in classical CNNs. The idea is to generate one feature map for each corresponding category of the classification task in the last convolutional layer. Then, instead of adding fully connected layers on top of the feature maps, we take the average of each feature map, and then feed the resulting vector directly into the SoftMax layer. [2]

To train the model we used the AdamW optimizer with a weight decay of 1e-5.

## 6.Results

### 6.1. Machine learning: 30s dataset

The results of Dhevan S. Lau et. Al. are reported in the table below.

| Classifier (30-second features) | Training Time | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| Logistic Regression | 487 ms | 66.0% | 66.50% |
| K-nearest Neighbours | 5 ms | 66.2% | 68.50% |
| Support Vector Machine | 73 ms | 66.5% | 73.50% |
| **Random Forests** | **5.72 s** | **69.3%** | **74.50%** |
| Multilayer Perceptron | 60.62 s | 62.9% | 67.50% |

*Table 1: Paper results with 30s features.*

The **Random Forests** technique is the most accurate method to classify the files based on the features sampled on the 30s audio file. The training time of this technique is one order of magnitude below the one of the Neural Network, but still achieves a higher accuracy.

The **K-Nearest Neighbors** method gives an interesting result: it is one of the best performing methods and it converges in just 5 milliseconds, overperforming models that take 10 to 10 thousand times more to be trained.

The results of most of techniques are still poor, 69% is an accuracy that is not suitable for any real application.

Our results are reported below.

| Classifier (30-second features) | Training Time | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| Logistic Regression | 83 ms | 71.7% | 69.0% |
| K-nearest Neighbours | 1 ms | 67.3% | 67.5% |
| **Support Vector Machine** | **25 ms** | **73.3%** | **70.0%** |
| Random Forests | 2.08 s | 70.4% | 68.0% |
| Multilayer Perceptron | 4.67 s | 71.1% | 71.5% |

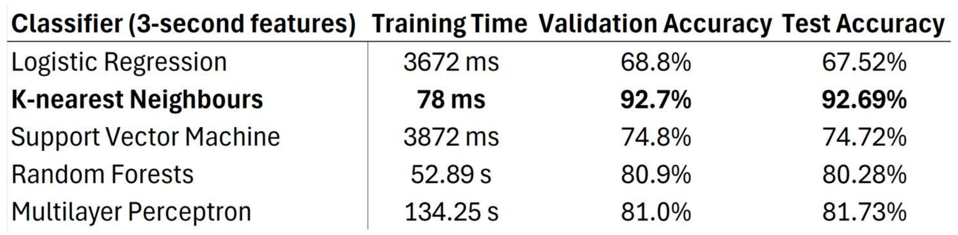
*Table 2: Out results with 30s features.*

Two things are particularly noticeable. The first one is that our training time is up to 5 times lower than theirs, indicating that we are working with a machine that is significantly more powerful. The second is that the different data preprocessing improved practically all the results (particularly noticeable in the multilayer perceptron) in terms of validation accuracy and favored other techniques rather than the Random Forests. The logistic regression, the Neural Network and the SVM are all overperforming Random Forests with the **Support Vector Machine coming on top** with an accuracy of 73.3% (4.0% improvement compared to their best result). Two models improved in terms of validation accuracy but worsened in terms of test accuracy.

The results are still not satisfactory. The size of the dataset is not sufficient for the training of the more complex techniques.

## 6.2. Machine learning: 3s dataset

The results of Dhevan S. Lau et. Al. are reported in the table below.

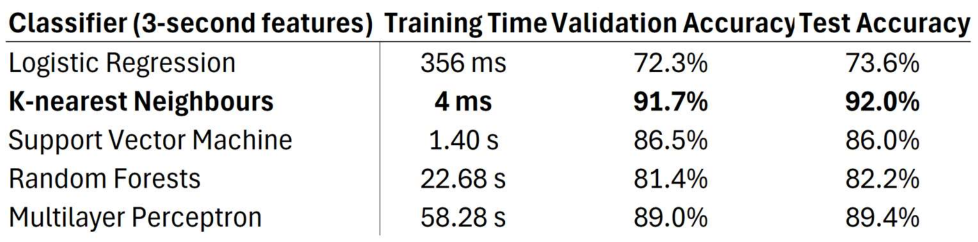


| Classifier (3-second features) | Training Time | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| Logistic Regression | 3672 ms | 68.8% | 67.52% |
| **K-nearest Neighbours** | **78 ms** | **92.7%** | **92.69%** |
| Support Vector Machine | 3872 ms | 74.8% | 74.72% |
| Random Forests | 52.89 s | 80.9% | 80.28% |
| Multilayer Perceptron | 134.25 s | 81.0% | 81.73% |

*Table 3: Paper results with 3s features*

**The increase in the size of the dataset appears to improve the accuracy** of all the machine learning methods. The **K-nearest Neighbors** method is the one that has the **highest accuracy** and the **lowest training time**, making it the obvious choice for this task.

Unlike the results with the 30s dataset, the results obtained with the 3s dataset are good enough to be used in a real-world application.



| Classifier (3-second features) | Training Time | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| Logistic Regression | 356 ms | 72.3% | 73.6% |
| **K-nearest Neighbours** | **4 ms** | **91.7%** | **92.0%** |
| Support Vector Machine | 1.40 s | 86.5% | 86.0% |
| Random Forests | 22.68 s | 81.4% | 82.2% |
| Multilayer Perceptron | 58.28 s | 89.0% | 89.4% |

*Table 4: Paper results with 3s features*

Again, we can observe again that our training time is significantly lower than theirs, and that the performance of most techniques has improved thanks to dataset normalization. Our results confirm that **K-nearest Neighbors** is the best approach for classification. And apparently the normalization does not provide an improvement in the accuracy of K-nearest Neighbors in this scope since our accuracy is 1% lower than theirs.

## 6.3. Neural Networks

| | Epochs | Test Loss | Test Accuracy |
|---|---|---|---|
| CNN (30-sec Features) | 30 | 1.609 | 53.50% |
| **CNN (3-sec Features)** | **50** | **0.873** | **72.40%** |
| CNN (Spectrograms) | 120 | 2.254 | 66.50% |
| NN (30-sec Features) | 150 | 0.249 | 82.50% |
| **NN (3-sec Features)** | **50** | **0.115** | **92.06%** |
| CNN (Spectrograms) | 120 | 0.294 | 78.50% |

*Table 5: Paper results above and our below*

The first half of the table shows the results obtained in the paper of Dhevan S. Lau et. Al., while the second half shows our results.

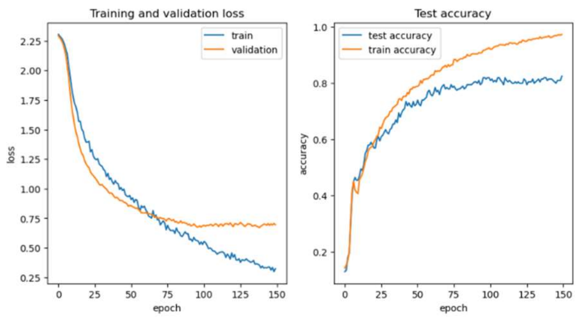
As said in the Methodology section we decided not to train Convolutional Neural Networks on features, since this model is designed to work with images. We decided to **create ad hoc Deep Neural Networks** instead and train it on the features datasets.

We start by analyzing the results of the **Convolutional Neural Networks trained on the spectrograms**. The original paper achieved a test accuracy of 66.5%. This model underperformed in terms of test accuracy compared to their machine learning models trained with 30s features (Random Forests 74.5%) and machine learning models trained with 3s features (K-nearest neighbors 92.7%).

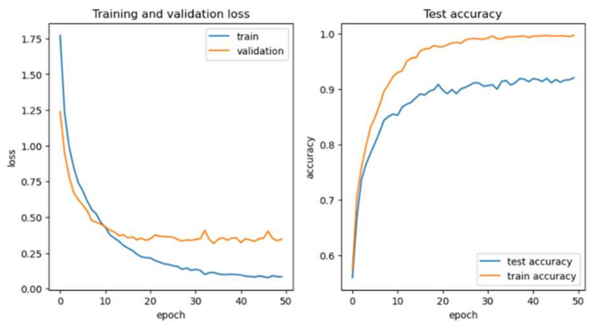
Adding a global average pooling layer has allowed us to increase the accuracy of the Convolutional Neural Network by 12% reaching a test accuracy of 78.5%. This model **outperformed all the machine learning models trained with 30s features** (Neural Network 71.5%), but the model still **underperformed compared to the same machine learning models trained with 3s features** (K-nearest Neighbors 92.0%), this can be explained by the effective size of the 3s dataset (10 times bigger).

Our **Deep Neural Network** outperformed their convolutional neural networks when trained on features. We achieved a test accuracy of 82.5% on 30s features (29% improvement) and a test accuracy of 92.1% (20% improvement) on 3s features. Our models are the **best performing in terms of test accuracy** on both 30s features dataset (82.5% compared to 71.5%) and 3s features dataset (92.1% compared to 92.0%), their convolutional models underperformed compared with Random Forests (53.5% compared with 74.5%) and K-nearest Neighbors (72.4% compared to 92.0%) respectively.

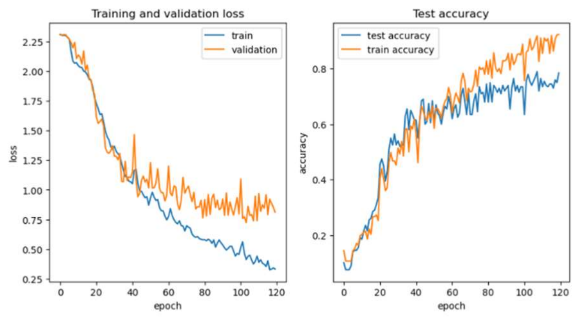
It is also worthy to mention that **the Deep Neural Network with 4 hidden layers performed better than the Multilayer Perceptron with 2 hidden layers** both with the 30s features dataset (82.5% accuracy compared to 71.5%) and the 3s features dataset (92% accuracy compared to 89.4%).



(a)



(b)



(c)

*Figure 2: Line graphs showing the accuracy and loss of training and validation for the Deep NN model trained on (a) 30-second feature set (b) 3-second feature set and CNN trained on (c) spectrograms.*

## 7.Conclusion

The comparative analysis shows some interesting results:

**Normalizing the dataset improves the performance** with almost every model. So, the *StandardScaler* should be preferred compared with the *MinMaxScaler* while preprocessing the dataset.

The spectrogram dataset and the 30s feature dataset have the same size, so it is reasonable to compare models trained with those two datasets. In this comparison the **Convolutional Model** would come up as the second-best performer (78.5% test accuracy), **outperforming all the machine learning models**. While the **best performer was the simple Deep neural network** with 82.5% accuracy.

On the 3s dataset **Neural Networks** can achieve an **accuracy that is comparable to one of the best performing machine learning models**, but at the cost of having a much **longer training time**. The K-nearest neighbors achieved an accuracy of 92.0% in only 5 milliseconds (running on the CPU only), while the Neural Network achieved the same result (92.1%) in more than 1 hour of training on the same CPU.
For real world applications, the improvement of performances does not justify the training time when compared to K-nearest Neighbors.

As suggested in the original paper the models trained on the **3s features dataset** performed better than the ones trained on the 30s one, this can be explained by the effective size of the dataset (10 times bigger).

Dhevan S. Lau and Ritesh Ajoodha decided to neither reject nor accept the hypothesis presented, we on the other end have found **sufficient evidence to accept the hypothesis**.
As we have seen above both the Deep and Convolutional Neural networks outperformed by a significative margin the classical machine learning techniques on the 30s dataset, while performed on par on the 3s dataset.

## 8.References

- Lau, D.S., Ajoodha, R. (2022). Music Genre Classification: A Comparative Study Between Deep Learning and Traditional Machine Learning Approaches.
- Gilbert Strang. (2019). Linear Algebra and Learning from Data. Wellesley-Cambridge Press.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville. (2017). Deep Learning. MIT press
- Qingkai Kong, Timmy Siauw, Alexandre M. Bayen. (2020). Python Programming and Numerical Methods. Academic Press
- Christopher M. Bishop. (2011). Pattern Recognition and Machine Learning. Springer
- Andrada Vulpe. Work w/ Audio Data: Visualize, Classify, Recommend. Kaggle
- Pytorch.org Documentation: https://pytorch.org/docs/stable/index.html
- [1] https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification
- [2] https://paperswithcode.com/method/global-average-pooling