Music Genre Classification using various Machine Learning and Deep Learning techniques

Tanmay Dokania
Dept. of Electrical Engineering
IIT Bombay
200070083@iitb.ac.in

Aziz Shameem
Dept. of Electrical Engineering
IIT Bombay
20D070020@iitb.ac.in

Kalp Vyas
Dept. of Electrical Engineering
IIT Bombay
kalp.vyas@iitb.ac.in

Nishant Thakre
Dept. of Electrical Engineering
IIT Bombay
200070051@iitb.ac.in

Abstract—**Music genre prediction is one of the topics that digital music processing is interested in. In this study, acoustic features of music have been extracted by using digital signal processing techniques and then music genre classification and music recommendations have been made by using machine learning methods. In addition, convolutional neural networks, which are deep learning methods, were used for genre classification and music recommendation and performance comparison of the obtained results has been. In the study, GTZAN database has been used and the highest success was obtained with the Random forest algorithm. Several machine learning approaches have been analysed and the process of tuning the hyper-parameters is elaborated on by trying a large set of parameters. Training and testing accuracies are used to ensure a model without overfitting or underfitting.**

Summary

In this implementation, we have used several Machine Learning / Deep Learning models, in an effort to classify music based on its genre. The data set used contains extracted features (57 in number), on which the training was performed, and the data was classified into 10 classes, representing different genres that the corresponding music belonged to.
The various models fitted are as follows:

- Decision Tree
- Random Forest Classifier
- Support Vector Machines
- Naive Bayes
- K - Nearest Neighbours
- Neural Networks

Feature Extraction

For each song, the raw music signal was divided into multiple features using a few digital processing techniques. Following are the features extracted:

- **Zero Crossing Rate:** no of sign changes of a signal in a certain period of time. For this there were 2 features, one for the mean and the other for the variance.
- **Spectral Centroid:** Centre of gravity of the frequencies in the frequency bin. For this there were 2 features, one for the mean and the other for the variance.
- **Spectral Bandwidth:** Weighted average amplitude difference between frequency magnitude and brightness. For this there were 2 features, one for the mean and the other for the variance.
- **Spectral Rolloff:** Frequency value corresponding to a certain ratio of the distribution in the spectrum. For this there were 2 features, one for the mean and the other for the variance.
- **Mel Frequency Coefficient of Cepstrum (MFCC):** Set of features which describe the overall shape of a spectral envelope. For this there were 20 sets of features used, each of which had 2 factors for mean and variance and hence a total of 40 features.

There are 9 other features in the GTZAN dataset which are chroma_stft (1 each for mean and variance, STFT: Short Time Fourier Transform), rms (1 each for mean and variance), harmony (1 each for mean and variance), perceptr(1 each for mean and variance) and tempo. This gives us a total of 57 features.

.

TABLE I
Feature Extraction

Feature	No of features
Zero Crossing Rate	2
Spectral Centroid	2
Spectral Bandwidth	2
Spectral Rolloff	2
chroma STFT	2
RMS	2
Harmony	2
Perceptr	2
Tempo	1
MFCC	40
Total	57

Now, lets look at the various models we have used to solve the problem of classifying a song into its genre based on its extracted features.

I. Decision Tree

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a data set, branches represent the decision rules and each leaf node represents the outcome.
The hyper-parameters tuned for maximizing performance are as follows :

- Criterion : gini or entropy
- Splitter : best or random
- Maximum depth of tree

The obtained results are summarized in Table 1. Note that trees with depth greater than about 10-11 tend to overfit the data. This can be seen from the plot of Accuracy Vs. Maximum Tree Depth, as shown in Figure 1.

TABLE II
Performance of Decision Tree for different hyper-parameters (Highlighted : Best performer)

Criterion	Splitter	Max. Depth	Train Accuracy	Test Accuracy
gini	best	8	62.1	55.4
gini	best	9	67.4	59.5
gini	best	10	73.3	62.9
gini	random	8	57.4	52.8
gini	random	9	63.2	53.6
gini	random	10	66.1	56.9
entropy	best	9	76.9	62.2
entropy	best	10	83.9	65.3
entropy	best	11	89.1	67
entropy	random	8	64.2	57.2
entropy	random	9	70.5	53.4
entropy	random	10	75.5	59.3

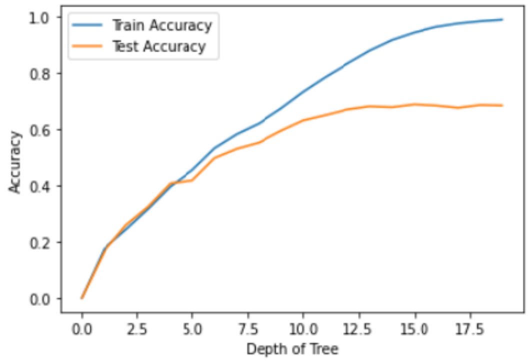

Fig. 1. Plot of Accuracy Vs. Maximum Tree Depth

II. Random Forest Classifier

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

The hyper-parameters tuned for maximizing performance are as follows :

- Number of Estimators
- Maximum Depth of Trees
- Criterion : Entropy or Gini

The results obtained are summarized in Table II. Note that the test accuracy obtained by Random Forest exceeds that obtained by a decision tree. This is expected, since Random Forests use several Decision Trees, and combine the results using Ensemble Methods.

TABLE III
Performance of Random Forest for different hyper-parameters (Highlighted : Best performer)

Criterion	No. Of Estimators	Max. Depth	Train Accuracy	Test Accuracy
entropy	50	3	47.4	45
gini	50	3	49.2	45.1
entropy	50	6	70.7	62.7
gini	50	6	69.9	64.4
entropy	50	9	94.5	78.4
gini	50	9	90.2	75.4
entropy	100	3	49.2	46.4
gini	100	3	51.6	48.5
entropy	100	6	71.5	64.1
gini	100	6	71.1	65.2
entropy	100	9	94.5	78.6
gini	100	9	90.3	76.2
entropy	200	3	47.8	45.1
gini	200	3	51	46.9
entropy	200	6	71.5	63.7
gini	200	6	71.3	63.4
entropy	200	9	95	79.8
gini	200	9	91.4	76.8

III. Support Vector Machines

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

Before fitting the model, we have employed dimensionality reduction using Principal Component Analysis(PCA).
PCA is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.
This makes it easier to fit the model, at the expense of accuracy of predictions.

The plot depicting the reconstruction error against the number of dimensions(resulting after PCA) is shown in Figure 2.

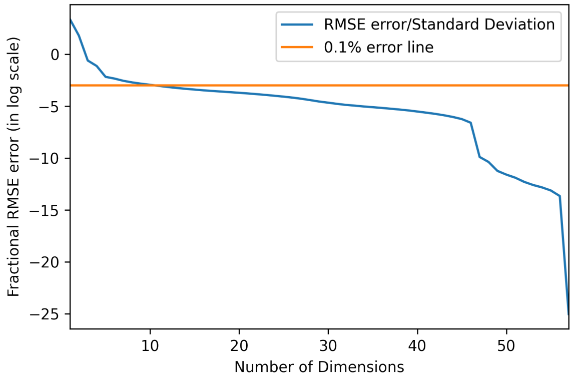

Fig. 2. Plot of Reconstruction error in log scale after PCA

The reconstruction error for having 10 dimensions is 0.0908% and hence, SVM is trained on this model. The following hyper-parameters were tuned during the training of SVM, on the data set

- C : Regularization parameter
- gamma : Kernel coefficient
- Kernel : Linear or rbf

The results obtained are summarized in Table III. The best performing set of hyper-paramters have been highlighted.

TABLE IV
PERFORMANCE OF SVM FOR DIFFERENT HYPER-PARAMETERS
(HIGHLIGHTED : BEST PERFORMER)

C	Gamma	Kernel	Train Accuracy
0.1	1	Linear	47.9
0.1	1	rbf	34.1
0.1	0.1	Linear	47.9
0.1	0.1	rbf	50.6
0.1	0.01	Linear	47.9
0.1	0.01	rbf	44.4
0.1	0.001	Linear	47.9
0.1	0.001	rbf	15.9
1	1	Linear	47.7
1	1	rbf	58.7
1	0.1	Linear	47.7
1	0.1	rbf	57.8
1	0.01	Linear	47.7
1	0.01	rbf	48.9
1	0.001	Linear	47.7
1	0.001	rbf	43.9

As compared to other models, the final accuracy obtaine dusing SVM is decent, considering that the training was performed on data compressed from 57 dimensions to 10 dimensions. This suggests that with enough computing power, SVM trained on the original data set could potentially out-perform the other algorithms.

IV. NAIVE BAYES

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. We have used 2 forms of naive Bayes classifiers namely, Gaussian and Bernoulli.

For Gaussian, the only hyper-parameter used was variance smoothing factor which was varies from 0 to 0.1

For Bernoulli, hyper-parameters used were :

- alpha: Additive (Laplace/Lidstone) smoothing parameter
- fit_prior : to decide whether to learn class prior probabilities or not

The results obtained are summarized in Table IV.

TABLE V
PERFORMANCE OF NAIVE BAYES FOR DIFFERENT HYPER-PARAMETERS
(HIGHLIGHTED : BEST PERFORMER)

Type	smoothing_factor	fit_prior	Test Accuracy
Gaussian	0.01	-	27.1
Gaussian	0.0001	-	28
Gaussian	0.000001	-	35
Gaussian	0	-	51.1
Bernoulli	0.01	True	29.4
Bernoulli	0.01	False	29.3
Bernoulli	0.0001	True	29.8
Bernoulli	0.0001	False	30.1
Bernoulli	0	True	31
Bernoulli	0	False	30.5

We can see that the Gaussian Naive Bayes algorithm clearly performs better than the Bernoulli one and gives a decent final accuracy of 51.1 percent. But amongst the other models, this model has a pretty low accuracy and is not a very good fit for this problem.

V. K-NEAREST NEIGHBOURS

The KNN (K-Nearest Neighbours) algorithm is one of the simplest algorithms in supervised ML and is mainly used for classification type of problems. It finds the distance between rows of test and train data using different criteria to measure distance like Hamming, Euclidean, Manhattan etc. It uses these distances to decide the class the set point belongs to. One interesting thing about this algorithm is that it always gives 100 percent accuracy on the train data so if the train data and test data are very similar, the algorithm works great.

The following hyper-parameters were tunes during training:

- N - no of neighbours being considered
- p - to decide which distance to be used (p=1 is manhattan (L1 norm),p=2 is euclidean(L2 norm))
- weights - to decide what weight functions to be used in prediction

We ran the hyper-parameters to vary N between all numbers from 1 to 100, p between 1 and 2 and weights were either uniform or distance. The best results were obtained on using N=34, p=1(manhattan distance) and weights = 'distance'. The test accuracy obtained was 32 percent which is the least compared to all the other methods.

VI. Neural Networks

Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

For the task, we have used a two-layer deep neural network, with several hyper-parameters tuned for optimizing performance.

These hyper-parameters include :

- Number of neurons in hidden layer
- Number of epochs
- Activation Function(used in hidden layers)
- Optimizers : Functions/Algorithms that modify the attributes of the NN, such as weights and learning rate

To decide the number of epochs to be trained, we have used the convergence of total loss on the validation data. When the validation loss stagnates, we end the training process. Any accuracy gained post this point is likely to have resulted from overfitting, and is not expected to be reflected in unseen data.

The result before and after scaling the data appropriately have also been shown, to highlight the importance of scaling. The accuracies in the two cases show a stark difference, both on train as well as test datasets.

The result of the training are summarized in the Table V and VI.

A. 3 second data

As is apparent from Table V, the maximum test accuracy was obtained using a two-hidden-layer deep network, containing 40 and 20 neurons, trained for 1000 epochs, using tanh as the activation function in both layers and Adam as the optimizer. Any other fit either led to high bias/underfit or high variance/overfit.

The plot of accuracies and losses against the number of epochs is shown in Figures 3 and 4 respectively.

B. 30 second Data

As is apparent from Table VI, the maximum test accuracy was obtained using a two-hidden-layer deep network, containing 40 and 20 neurons, trained for 1286 epochs, using sigmoid as the activation function in both layers and Adam as the optimizer.

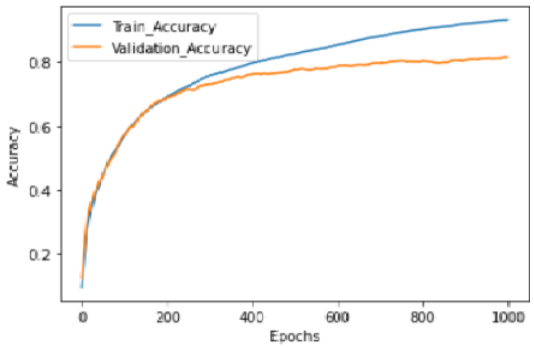

Fig. 3. Plot of Model Accuracy Vs No of Epochs

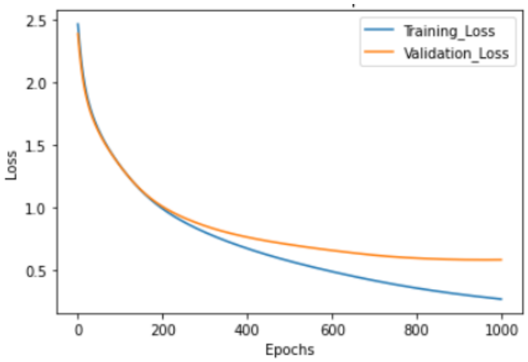

Fig. 4. Plot of Loss Vs No of Epochs

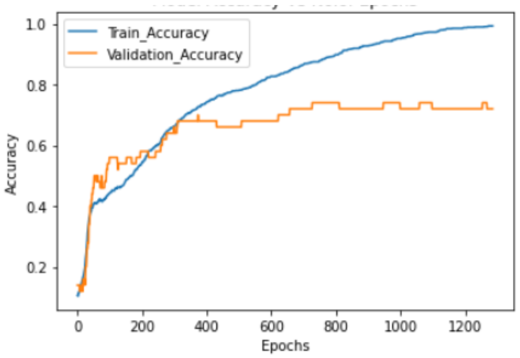

Fig. 5. Plot of Model Accuracy Vs No of Epochs

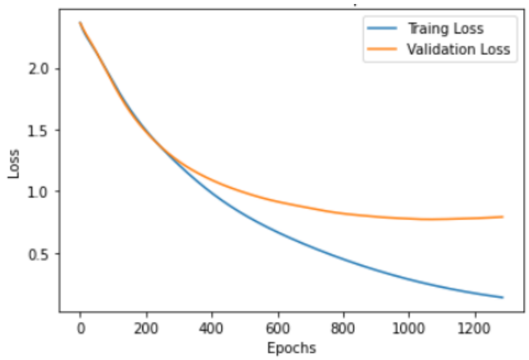

Fig. 6. Plot of Loss Vs No of Epochs

TABLE VI
PERFORMANCE OF ANN : 3 SECOND DATA

Number of Hidden Layers	Scale	Number of Neurons in hidden layer	Number of Epochs	Activation Functions	Optimizer	Train Accuracy	Test Accuracy
2	No	40, 20	157 (Conv)	Relu, Relu	Adam	16.15%	17.10%
2	Yes	40, 20	752 (Conv)	Relu, Relu	Adam	91.05%	80.04%
2	Yes	40, 20	400	Relu, Relu	Adam	82.73%	75.30%
2	Yes	40, 20	600	Relu, Relu	Adam	88.64%	79.60%
2	Yes	40, 20	600	Relu, Relu	SGD	47.51%	47.30%
2	Yes	40, 20	600	Relu, Relu	RMSprop	89.22%	79.40%
2	Yes	40, 20	292 (Conv)	Relu, Relu	RMSprop	82.74%	76.20%
2	Yes	40, 20	1000	Tanh, Tanh	Adam	93.07%	81.10%
2	Yes	30, 20	2000	Tanh, Tanh	Adam	96.13%	78.40%
2	Yes	40, 20	1000	Sigmoid, Sigmoid	Adam	79.16%	74.00%

TABLE VII
PERFORMANCE OF ANN : 30 SECOND DATA

No. of Hidden Layers	Scale	No. of Neurons in each hidden layer	Number of Epochs	Activation Functions	Optimizer	Train Accuracy	Test Accuracy
2	No	40, 20	850	Relu, Relu	Adam	22.00%	22.00%
2	Yes	40, 20	1000	Relu, Relu	Adam	99.89%	62.00%
2	Yes	40, 20	285 (Conv)	Relu, Relu	Adam	91.89%	58.00%
2	Yes	40, 20	1000	Relu, Relu	SGD	63.89%	60.00%
2	Yes	40, 20	2000	Relu, Relu	SGD	79.89%	58.00%
2	Yes	40, 20	157 (Conv)	Relu, Relu	RMSprop	85.67%	58.00%
2	Yes	40, 20	1286 (Conv)	Sigmoid, Sigmoid	Adam	97.67%	70.00%
2	Yes	40, 20	515 (Conv)	Tanh, Tanh	Adam	98.78%	60.00%
2	Yes	30, 20	1464 (Conv)	Sigmoid, Sigmoid	Adam	98.44%	64.00%
2	Yes	30, 20	1000	Sigmoid, Sigmoid	Adam	90.33%	64.00%

Any other fit either led to high bias/underfit or high variance/overfit(as can be seen from the large gap between train and test accuracies).

The accuracy obtained in this case is lesser than that obtained on the 3-second data. This is expected, given that the number of data points in this case is far lesser than that in the previous case.

The plot of accuracies and losses against the number of epochs is shown in Figure 5 and 6 respectively.

VII. CONCLUSION

Table VII summarizes the accuracies obtained from the models fitted on the data :

Based on the result obtained, we can see that Neural Networks have given the best result in terms of accuracy, however, Random Forests are not far behind, giving similar accuracies with much lower training times.

CONTRIBUTIONS

The publication and data searching was done by Tanmay and Kalp. Pre-processing of the data was done by Kalp.

Out of the six models trained, the distribution of work was as follows :

- Nishant Thakre : DT, RF
- Tanmay Dokania : SVM (PCA)
- Kalp Vyas : NB, KNN
- Aziz Shameem : ANN

The report drafting and compilation was accomplished jointly by Aziz, Kalp and Tanmay.

REFERENCES

[1] ”Music Genre Classification and Recommendation by Using Machine Learning Techniques” *https://ieeexplore.ieee.org/document/8554016*

[2] ”Feature Extracted Data sets” *https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification*

.

TABLE VIII
FINAL ACCURACIES

Model	Accuracy
Decision Tree	67
Random Forest	79.8
Neural Nets	81.1
Naive Bayes	51.1
SVM	58.7
KNN	32