Machine Learning Report

Qiaotong Huang, Gulinigeer Aihemaiti
College of Engineering
Northeastern University
Toronto, ON
huang.qiaot@northeastern.edu
aihemaiti.gu@northeastern.edu

Abstract

Music is one of the earliest borderless languages, and audio data is becoming an important source of information in machine learning. One readily available source of audio data is music recordings. Countless music lovers hope that artificial intelligence will create better applications in the field of music. In recent years, deep learning has achieved good success in various tasks such as music genre classification, music recommendation, and music generation. In this report, we explore several different machine learning architectures for classifying music audio files into ten different genres. By extracting MFCC[1] feature values from music files and using different models for modeling. We present the best results of each model and compare them.

1 Introduce

Music genre classification is a popular problem in machine learning with a variety of applications. It can be used to label each song in a huge corpus of music by genre and sub-genre, which can then be used to identify similar songs. Another application is a music recommendation system. By using features from the middle layer in the model, we can cluster similar songs and share these songs with users based on their preferences.

Traditionally, we extract features from music audio such as MFCC and use them as a starting point for classification tasks. In this project, I first convert each audio music file into a mel spectrogram. A spectrogram is a visual representation of the amplitude of different frequency bands over time. Different types of spectrograms also look very different. Then, we tested different machine learning algorithm models and showed their results. Finally, we can treat the spectrogram as an image and use it as input to the convolutional recurrent model, where the convolutional layer and the recurrent layer Extract features from spectrograms in parallel or sequentially to perform classification. This report is divided into several different parts - Part 1 defines the problem, Part 2 describes the dataset and generation of the mel spectrogram, Part 3 covers the model architecture we used and its results, and we present it in Part 4 Conclusions are given in Sect.

2 Approach

In this section, I will talk about our choice of data set and how audio files are converted into a spectogram.

2.1 Choice of Dataset

We used a dataset that was was taken from the Kaggle[2]: it is open source and open to anyone to use, in particular in academia, but also in the industry. Our initial data consisted of 1000 .au files (a file extension used for audio files), split in 10 different music genres (Disco, Metal, Pop, Hip Hop, Rock, Blues, Classical, Country, Jazz, Reggae), with 100 samples of 30 seconds for each genre.

Our next step was to find a way to convert this raw data to a format that our machine learning techniques could use. We were referred to the use of [3]: Mel Frequency Cepstral Coefficients.

2.2 Mel Spectrogram Generation

Each audio file was converted into a spectogram which is a visual representation of spectrum of frequencies over time. A regular spectogram is squared magnitude of the short term Fourier transform (STFT) of the audio signal. This regular spectogram is squashed using mel scale to convert the audio frequencies into something a human is more able to understand. I used the built in function in the librosa library to convert the audio file directly into a mel-spectogram. The most important parameters used in the transformation are - window length which indicates the window of time to perform Fourier Transform on and hop length which is the number of samples between successive frames. The typical window length for this transformation is 2048 which converts to about 10ms, the shortest reasonable period a human ear can distinguish. The hop length of 512 was chosen. Further more the Mel-spectrograms produced by Librosa were scaled by a log function. This maps the sound data to the normal logarithmic scale used to determine loudness in decibels (dB) as it relates to the human-perceived pitch. As a result of this transformation each audio signal gets converted to a mel-spectogram of shape - 640, 128.

It can be clearly seen from the figure that there are significant differences in the spectrograms of different styles of music. This is why we can determine the classification of music by analyzing this extracted data set.

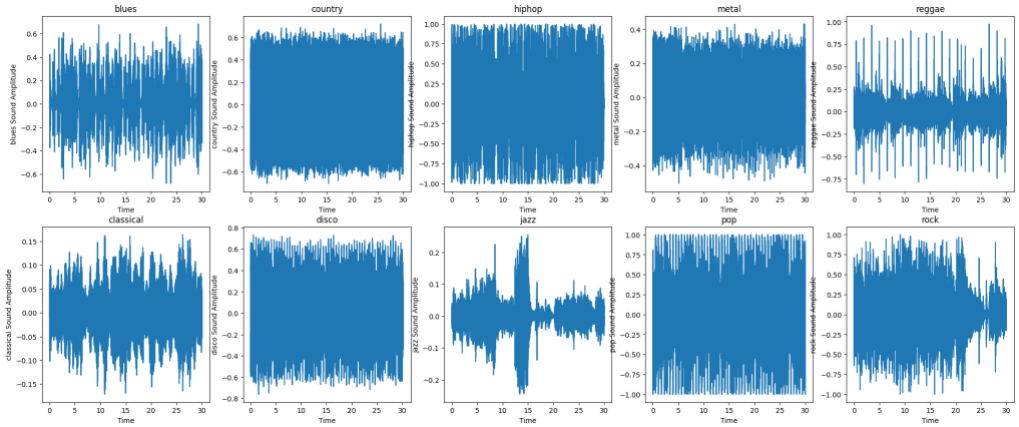
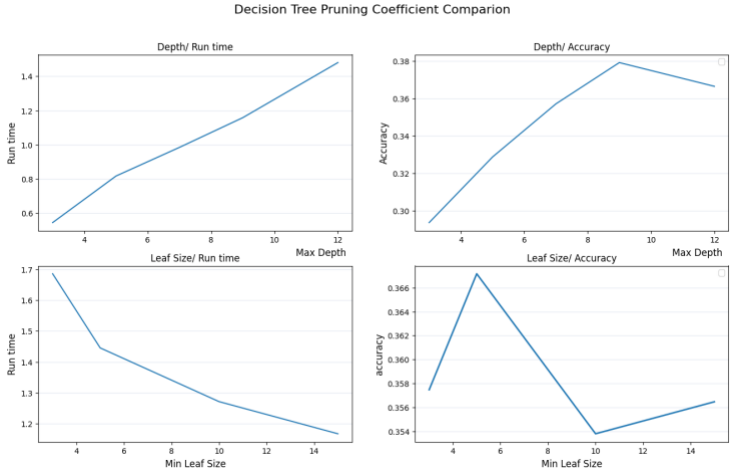


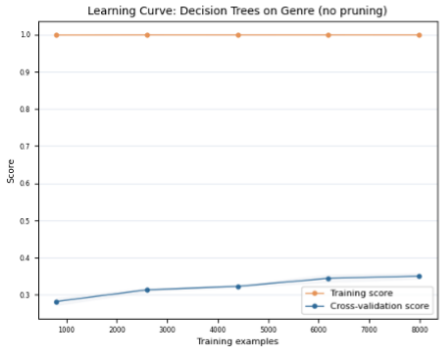
Figure 1: Comparison of spectrograms of different styles of music

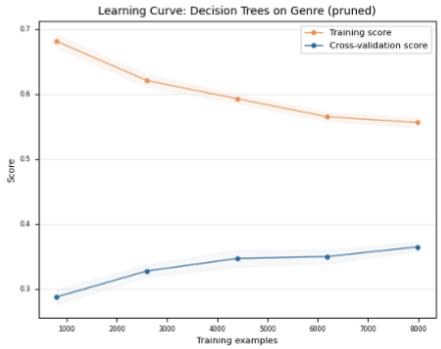
3 Modeling Approach and Results

3.1 Decision Tree

First test different tree depths and number of leaves, and then find the best parameters among them: tree depth is 9, number of leaves is 5. Then use this parameter to train the model. You can see that as the number of test cases increases, the accuracy of the decision tree improves significantly, and pruning can significantly reduce the over-fitting of test cases. The accuracy without pruning is 34.75%, and the accuracy after pruning is 35.08%.

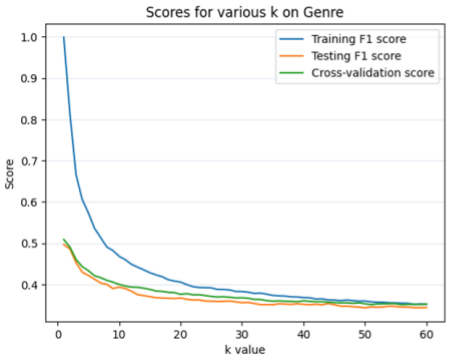


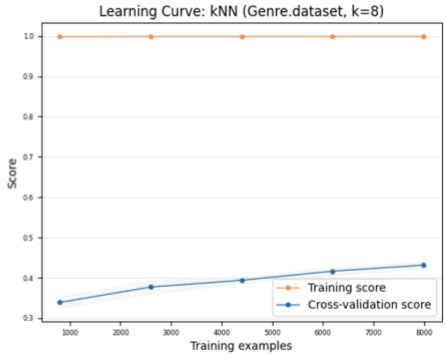




3.2 kNN

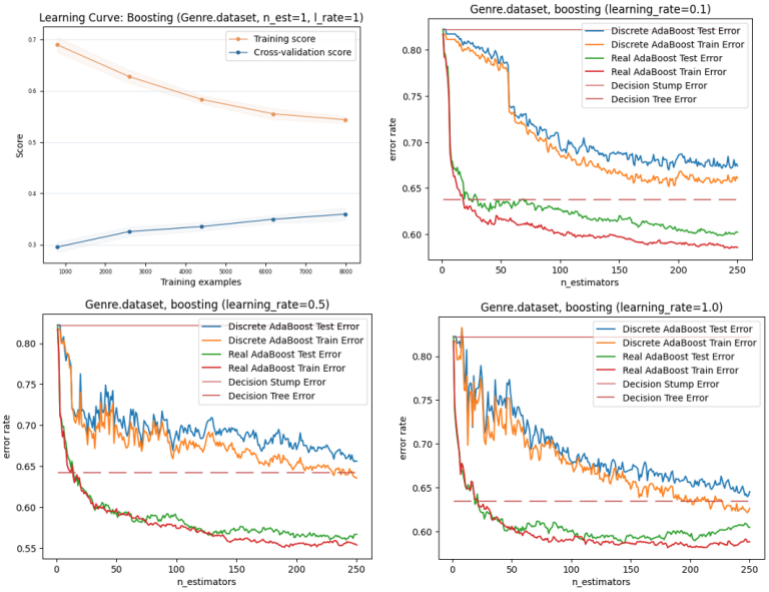
To test k between 1 - 60, we need to weigh the accuracy and sample deviation, and finally we choose k=8 for model training. As the number of samples increases, the accuracy of the model has been steadily improving. The final accuracy is about 40%.





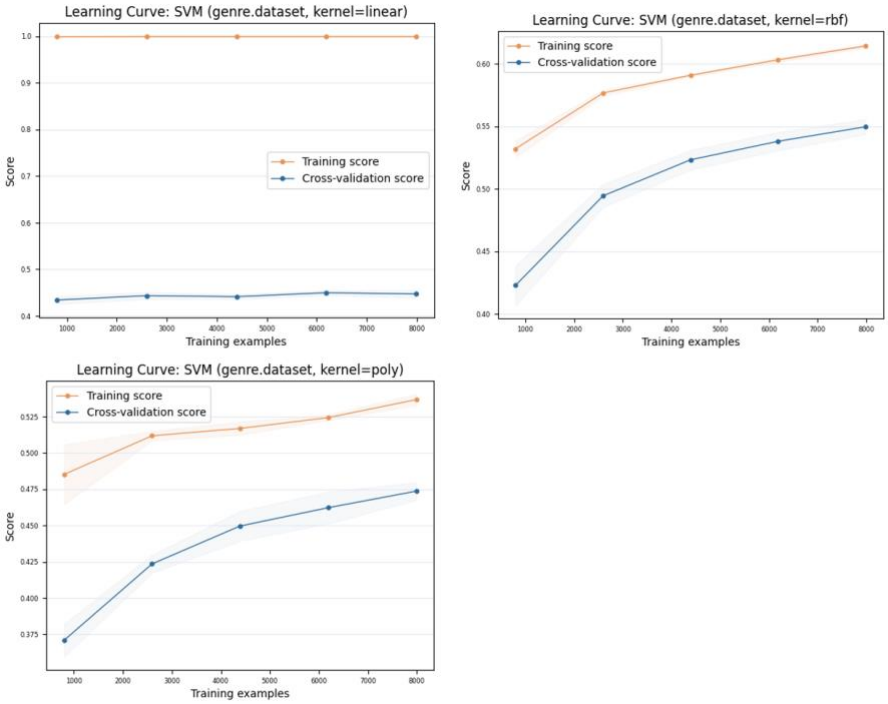
3.3 Boosting

Different learning_rate were compared in Boost, and different amounts of data were compared when learning_rate=1.0. It can be found that as the amount of data increases, the training accuracy decreases, the test accuracy increases, and finally the test accuracy is obtained. is 34.01%.



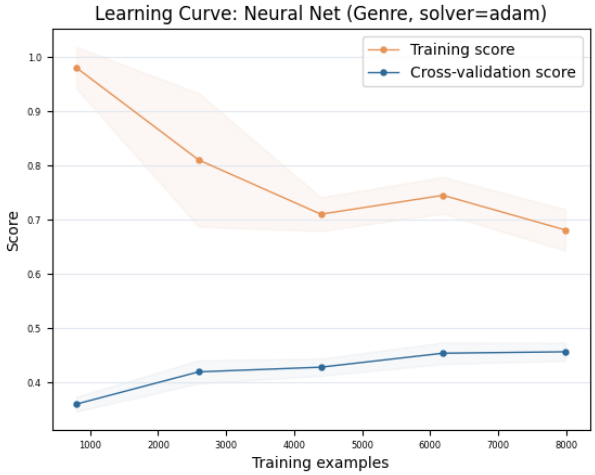
3.4 SVM

Using three different kernel functions such as linear, rbf, and poly in SVM for comparison, it can be found that linear is the least sensitive to the amount of data, and the accuracy of the other two functions has significantly improved as the amount of data increases. The final accuracy rate of linear is 42.65%, the accuracy rate of rbf is 54.27%, and the accuracy rate of poly is 50.51%. Obviously, the rbf kernel function works best.



3.5 MLP

In MLP, we selected adam as the solver and studied the relationship between the amount of training data and accuracy. It can be found that as the training data increases, the accuracy of cross-validation increases significantly, but when the training data reaches more than 6000, due to The optimization effect brought by the amount of data is no longer obvious. The final accuracy rate was 47.39%.



3.6 Artificial Neural Networks (ANN)

Overfitting occurred during the initial training of the ANN (Figure 2). We can mainly deal with it through the following process: 1. Making architecture less complicated; 2. Using augmented data; 3. Early stopping of training; 4. Adding dropout layers; 5. Regularization / Standardization.

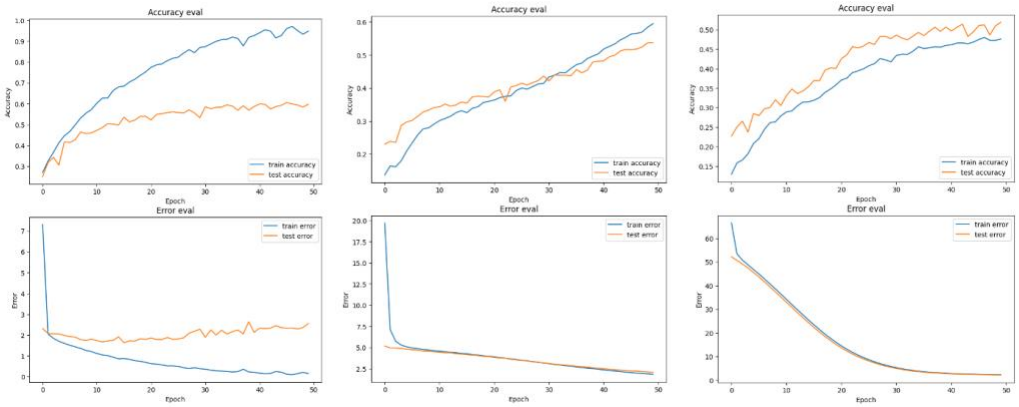


Figure 2: Learning curve of ANN model (left: before optimization, middle: L2 optimizationm, right: L1 optimization)

Compared to the previous naive model, we added a dropout layer and kernel_regularizers, giving a dropout probability of 30%. Kernel_regularizers is one of the 3 regularizers used to impose penalties, and compared L1 regularization and L2 regularization. After improvement, the overfitting phenomenon of the model was significantly reduced (Figure 2). It can be seen that the error rate drops faster after L2 regularization, and the accuracy is also better than L1 regularization.

3.7 Convolutional Neural Networks (CNN)

We can see that although the overfitting of the ANN is greatly reduced after adjustment, we still cannot obtain good accuracy. Now we will try to do this using Convolutional Neural Networks (CNN). CNN are commonly used in image recognition related tasks. They perform convolution operations instead of matrix multiplications and are often used in early layers to understand the two-dimensional layout of data. The hierarchical structure of the neural network model we build here is linear, stacked layer by layer. A flattening layer is added to the model to flatten the feature map into a one-dimensional vector for connection to the fully connected layer. A fully connected layer is added, containing 64 filters, using the ReLU (Rectified Linear Unit) activation function. A dropout layer is added to prevent overfitting. It discards some neurons with probability 0.3. The last line adds a fully connected layer with 10 neurons as an output layer using a softmax activation function for multi-class classification problems.

We use keras layers of Conv2D, MaxPool2D, Batch Normalization. The CNN layer mainly accepts inputs of 3D shapes, so again we have to prepare the dataset in form, for this we use the np.newaxis function which adds columns/layers in the data.

The model was trained using the Adam optimizer. After 50 epochs of training, the verification accuracy gradually stabilized at 70%, which is the best learning effect achieved by the model under this parameter. As shown in the figure, the test results and training results of this model have the highest consistency before and after 20 epoch training. This means that even after using multiple regularization techniques, if the training epochs are excessive, the model is somewhat overfitted for training and does not result in significant improvements in test accuracy.

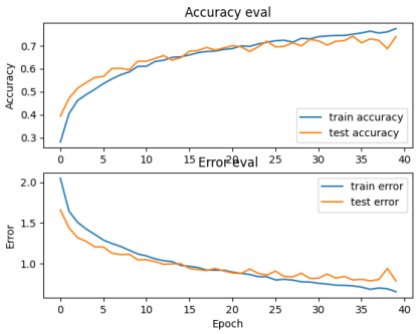


Figure 3: Learning curve of CNN model

4 Conclusion and Future Work

Experiments so far show that deep learning models using CNNs can perform better than all other algorithms, showing the unique advantages of convolutional neural networks in music processing. This also proves that deep learning itself can extract useful features from the original mel spectrogram. Additionally, while the accuracy of the model is lower, we believe this may be due to two things : 1. Insufficient samples. Even 1000 spectrograms per type is probably a small sample here since we are training these models from scratch. A larger data set should improve the results. 2. Challenges of GTZan dataset and confusion between categories. There is a paper on dataset errors [6] that mentions duplication, mislabeling, and distortion. For future work:

I will try more data sets. Including additional public data on Kaggle, as well as relying on the publicly available Spotify API, which allows us to capture information about specific songs, artists, and their albums. Of course, how to provide data through the public API of the music platform requires a lot of data processing, including labeling the songs appropriately.

References

[1] Wikipedia "MFCC" https://en.wikipedia.org/wiki/Mel-frequency_cepstrum

[2] Andrada "GTZAN Dataset - Music Genre Classification". https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification

[3] Practical Cryptography "Mel Frequency Cepstral Coefficient (MFCC) tutorial". http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

[4] Deep Learning for Music Genre Recognition (Priyanka Dwivedi, 2018)

[5] Music Genre Classification (Matthew Creme, Charles Burlin, Raphael Lenain, 2016)

[6] The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use (Bob L. Sturm, 2013)