Music Genres classification via Short-time Fourier transform

Muhammad Faique
Department of Engineering Science
University at Buffalo
Buffalo, NY 14260
mfaique@buffalo.edu

Yinxia Chen
Department of Engineering Science
University at Buffalo
Buffalo, NY 14260
yinxiach@buffalo.edu

Xueping Leng
Department of Computer Science
University at Buffalo
Buffalo, NY 14260
xleng@buffalo.edu

Abstract

This project is designed to use a short-time Fourier transformation of the data to classify the genres of the music instead of directly training with the audio itself. The transformed data is then trained with a CNN model and an FNN model. Several metrics evaluate the model's performance, and we use the FNN model that is evaluated by other features to compare and measure the effectiveness and accuracy of the deep learning model. We also introduced a KNN to evaluate if such transformation can provide a clear view of the characters that can be used to classify the music. The models described in this project can be used to classify the genres of the music using a method that is different from the original way of training the audio data itself and can be used to develop further models to help look for the clear difference between different genres of music.

1 Introduction

Music is a part of nearly everyone's life. Every person has their own set of songs on their playlist. Every person likes different kinds of music, and it has different meanings. It is stated that "Research has found that when a subject listens to music that gives them the chills, it triggers a release of dopamine to the brain" [6].

There exist different music genres, which is a significant way for most people to classify different types of music. Although differences exist among other genres, the edge seems not to be cleared. Most of the differences are introduced as subjective emotions and mood or by the history that is not sensitive to the music itself. Hence, to study the difference between different genres in the objective sense, we would like to introduce deep learning models to classify the genres and seek some patterns that can be hard to be realized by a human ear.

2 Motivation

The music genre classification problem is a classical classification problem that has already been introduced for years. However, most of the models applied to the model are presented based on the audio model itself.

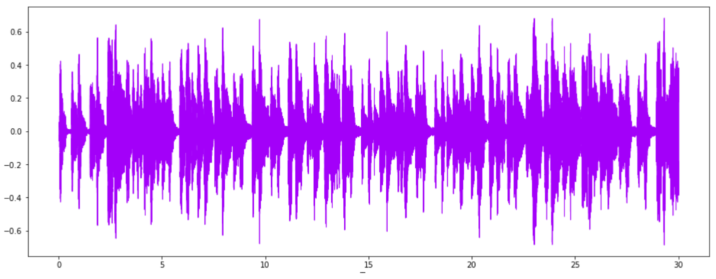


Figure 1. The wave plot of an audio file. This provides visualization of the audio data that most of the classification models used.

However, most of the models cannot offer an explainable production as a human cannot understand and categorize a wave plot. Therefore, we tried to use a Short-time Fourier Transformation (STFT) to the audio data, trying to feature and weight the audio more efficiently while making a more readable and explainable outcome.

3 Short-time Fourier Transformation

The Short-time Fourier transform (STFT), is a Fourier-related transform used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time.[1]

The formula of the STFT is

$$X_w(mS,\omega) = F_1[X_w(mS,l)] = \sum_{l=-\infty}^{\infty} X_w(mS,l)e^{-j\omega l} \text{ ,} where\ X_w(mS,l) = w(mS-l)x(l)$$

Where x_n denotes a real sequence and $X_w(mS,\omega)$ indicate it's STFT. S is a positive integer of the sampling period of $X_w(mS,\omega)$ in n. The analysis window used in the STFT is denoted by $w(n)$ with its little loss of generality L, which limits the n by $0 \leq n \leq L-1$. [2]

We use a python package, librosa, to perform the transformation. It is a package for music and audio analysis and provides the building blocks necessary to create music information retrieval systems. [3][4].

Once we finished the transformation, we can create a spectrogram based on the transformation result.

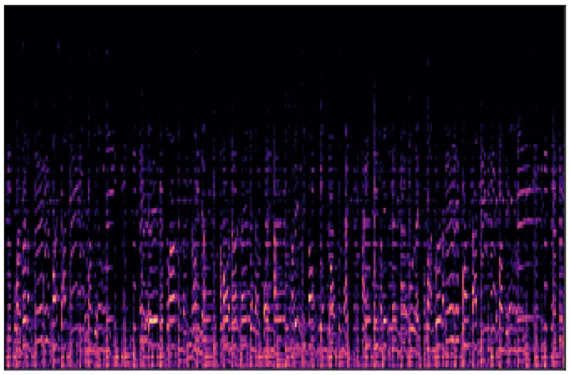


Figure 2. The spectrogram of an audio data after STFT

Such spectrogram can reduce the influence of the audio length and time tag and precent the features based on the characteristics of the sound. It would give a more explainable diagram.

4 Data

The data we used to train the model comes from Kaggle. [5] It contains collections of 10 genres with 100 audio files each, all having a length of 30 seconds. The data is collected from GTZAN dataset, which is the most-used public dataset for evaluation in machine listening research for music genre recognition.

It also includes some files that conclude the mean and variance of features of the music in the dataset such as spectral centroid and spectral bandwidth.

5 Models

5.1 K-Nearest Neighbors Model

The first model we trained for comparison purposes is K Nearest Neighbors trained with the extracted feature dataset. In order to determine the genres of the music, KNN algorithm allows us to label a unlabeled data through calculating its distance with other labeled data, find the most K closest labeled data therefore label the test data. In our model, we weighted neighbors with their distance from the unlabeled test data point. Thus, the closer neighbors have the higher weight.

In our implementation, we directly use KNeighborsClassifier from sklearn. Our data is separated using train_test_split method provided by sklearn, so the training data and testing data has high randomness. In such a case, we guarantee that our KNN model provides expected performance by determining the optimal K value based on the minimum of RMSE value rather than setting a fixed value for K. The lookup K value range we are setting is [1, $N^{1/2}$] where N is the size of training data frame since some research found that the best performance K value could be up to approximate $N^{1/2}$ in terms of different dataset [6].

The RMSE value with the different K value is shown in the following figure:

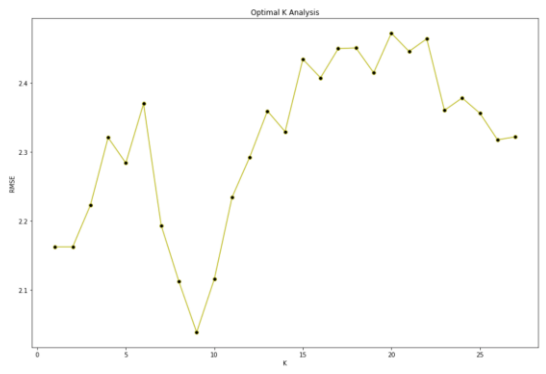
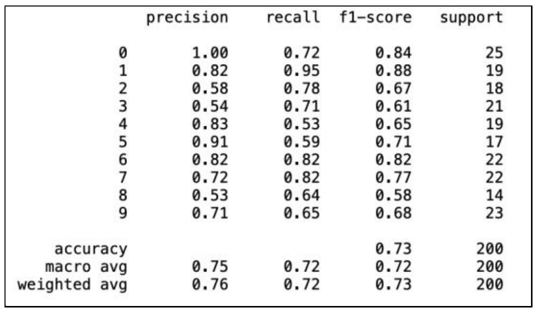


Figure 3. Relationship between K value and RMSE

The accuracy we achieved for KNN model is around 70%, as the following figure shows:



```
              precision    recall  f1-score   support

           0       1.00      0.72      0.84        25
           1       0.82      0.95      0.88        19
           2       0.58      0.78      0.67        18
           3       0.54      0.71      0.61        21
           4       0.83      0.53      0.65        19
           5       0.91      0.59      0.71        17
           6       0.82      0.82      0.82        22
           7       0.72      0.82      0.77        22
           8       0.53      0.64      0.58        14
           9       0.71      0.65      0.68        23

    accuracy                           0.73       200
   macro avg       0.75      0.72      0.72       200
weighted avg       0.76      0.72      0.73       200
```

Figure 4. Detail of KNN Model Performance

In the following section, we explore another two models which are expected to bring better performance.

5.2 Feedforward Neural Network Model

The second model we used is feedforward neural network which is a supervised model. We use Keras to implement our neural network. Our FNN model is fed with 800 labeled training data, and then label 200 testing data. It has 6 layers and the detailed information for each layer is shown in the following table:

Table 1: FNN Model Architecture

layer	Activation Function	Output Shape	Parameter Number
1	ReLU	(None, 512)	30208
2	Batch Normalization	(None, 512)	2048
3	ReLU	(None, 256)	131328
4	ReLU	(None, 128)	32896
5	ReLU	(None, 64)	8256
6	Softmax	(None, 10)	650

In our dataset, there are ten genres of music: blue, classical, country, disco, hip-hop, jazz, metal, pop, reggae, rock. They are encoded into different values for the purpose of data being labeled. The output of the model is a vector with ten values. Each value indicates the possibility of the data point belongs to a certain music class. Finally, when we are analyzing the model, the vector will be transferred into one-hot form to indicate the class that the music is classified into.

For analyzing our training performance, categorical cross entropy is used, the objective formula is:

$$L = -\sum_{i=1}^{p} y_i \log \tilde{y}_i$$

where p is output size, in our case p = 10.

The accuracy we achieved for this model is around 80% with the loss trending as following:

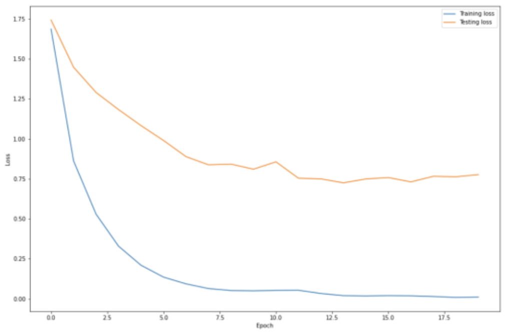
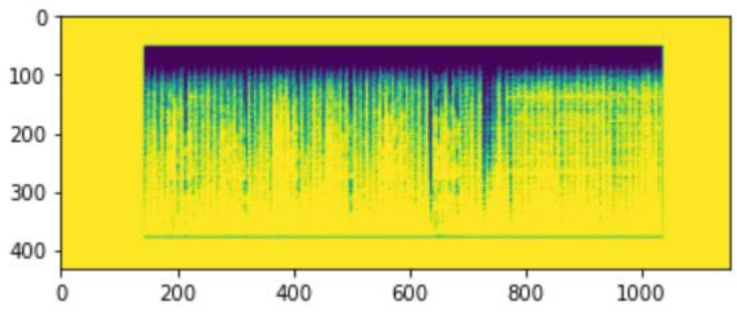


Figure 5. Trend of Training Loss and Testing Loss

Since we use the feature data extracted from the audio files, the data can lose some information about the music in terms of time dimension. Thus, we used the spectrogram transformed from the raw audio to train our CNN model.

5.3 Convolution Neural Network Model

The third model which we use is CNN[7]. For this model spectral images audio data, which was converted from audio, as described in above section. After that, these images have been converted to gray scale in order to reduce dimensions. Then every image is passed through a vertical edge filter. All of the images then look like:



Then every image is normalized after dividing these by 255. After this further preprocessing, we designed our convolution neural network [7]. Following is the construction of our network.

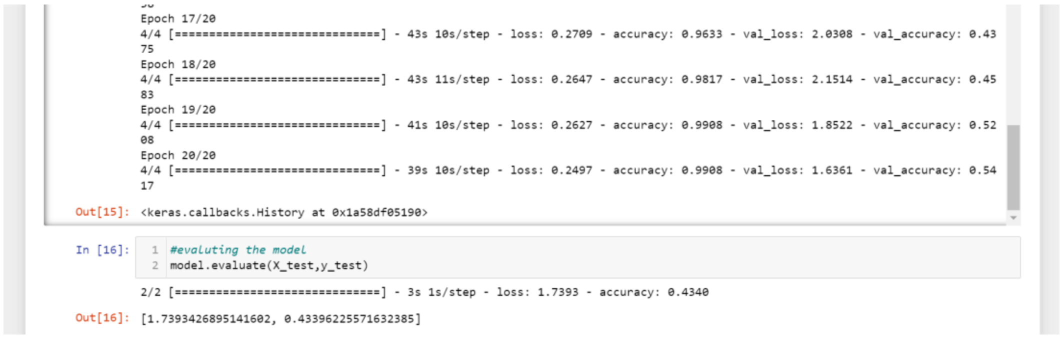
a) Convolution Layer 1: Our layer after input layer is convolution layer. This layer has 16 kernels of size 5x5 with stride=1 and padding=2. This layer take input of size 432x1152x1 and return output of size 432x1152x16. This layer has ReLU as an activation function.
b) Pooling Layer 1: Our second layer is a pooling layer with F=2x2 and Stride=2. It gives us output of size 216x576x16.
c) Convolution Layer 2: Our layer after input layer is convolution layer. This layer has 32 kernels of size 5x5 with stride=1 and padding=2. This layer takes input of size 216x576x16 and return output of size 216x576x32. This layer has ReLU as an activation function.
d) Pooling Layer 1: Our second layer is a pooling layer with F=2x2 and Stride=2. It gives us output of size 108x288x16.
e) Fully Connected Layer: Next layer is fully connected layer with ReLU function and L1 and L2 regularizes.
f) Output Layer: Last one is output layer which has SoftMax function. It returns probability of all the ten classes.

This model also contains early stopping optimizer. This model has validation set which comprise of 30% of training set.

Highlights:

- Initial this CNN was designed without regularizer and optimizer, which cause model to over fit. It was reaching training accuracy of 1 while test accuracy was as low as 0.2.
- We then introduced L1 and L2 regularizer. After that this overfitting problem was resolved but model still unable to became generalized.
- We are only using 20% of our total data due to capacity of our computers. We believe that our model will perform better if full data is used.

Results:



```
Epoch 17/20
4/4 [==============================] - 43s 10s/step - loss: 0.2709 - accuracy: 0.9633 - val_loss: 2.0308 - val_accuracy: 0.43
75
Epoch 18/20
4/4 [==============================] - 43s 11s/step - loss: 0.2647 - accuracy: 0.9817 - val_loss: 2.1514 - val_accuracy: 0.45
83
Epoch 19/20
4/4 [==============================] - 41s 10s/step - loss: 0.2627 - accuracy: 0.9908 - val_loss: 1.8522 - val_accuracy: 0.52
08
Epoch 20/20
4/4 [==============================] - 39s 10s/step - loss: 0.2497 - accuracy: 0.9908 - val_loss: 1.6361 - val_accuracy: 0.54
17

Out[15]: <keras.callbacks.History at 0x1a58df05190>

In [16]: #evaluting the model
         model.evaluate(X_test,y_test)

2/2 [==============================] - 3s 1s/step - loss: 1.7393 - accuracy: 0.4340

Out[16]: [1.7393426895141602, 0.43396225571632385]
```

Accuracy 0.4340 on 20 Epochs.

6 Summary

Compared with the performance of the KNN and FNN models, the result we got from the CNN model is not ideal. It seems like the overfitting issue has not been completely eliminated even after we introduced the early stopping optimizer and regularizer because the training accuracy has achieved 99% while the testing accuracy is only 43%.

Based on the result we get from our three models; our dataset fits better with the extracted feature data. However, due to the computational limit of our devices, our CNN model is only trained with 20% of the whole dataset. After STFT, the dimension of input is higher resulting in a high computation training process. Hence, we believe the performance of our CNN model can still be improved.

7 Discussion

After optimizing our model on further data, we would be able to successfully classify the genre of more music audio. Classifying the audio via STFT may provide a more explainable classification outcome, as we believe the spectrogram is more interpretable than the wave plot, therefore may help in further research on the characteristic of music genres.

Further development in this type of classification may also bring a different chance to a music recommendation system that is not based on the human-classified genres, but on the characteristic of the music. This may provide better recommendation values than the genres by grabbing the specific characteristic that users preferred and providing them with music that has a similar characteristic. Our next focus will be adding users and their preferred choices of music in past. After carefully classifying their past choices of audio, we can aim for recommending different music.

8 Source Code

https://github.com/yinxiach/CSE676-music-genre-classification.git

References

[1] Sejdić E.; Djurović I.; Jiang J. (2009). "Time-frequency feature representation using energy concentration: *An overview of recent advances". Digital Signal Processing.* 19 (1): 153–183. doi:10.1016/j.dsp.2007.12.004.

[2] Griffin, D., &amp; Lim, J. (1984). Signal Estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2), 236–243. https://doi.org/10.1109/tassp.1984.1164317

[3] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference* (Vol. 8).

[4] Brian McFee, Alexandros Metsai, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Frank Zalkow, Ayoub Malek, Dana, Kyungyun Lee, Oriol Nieto, Dan Ellis, Jack Mason, Eric Battenberg, Scott Seyfarth, Ryuichi Yamamoto, viktorandreevichmorozov, Keunwoo Choi, Josh Moore, … Thassilo. (2022). librosa/librosa: 0.9.1 (0.9.1). Zenodo. https://doi.org/10.5281/zenodo.6097378

[5] Andrada Olteanu. (2020). GTZAN Dataset - Music Genre Classification, version 1. Retrieved May 2022, from https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification.

[6] Paryudi, I. (1970, January 1). What affects K value selection in K-nearest neighbor: Semantic scholar. Retrieved May 22, 2022, from https://www.semanticscholar.org/paper/What-Affects-K-Value-Selection-In-K-Nearest-Paryudi/34c2817df22843d5dc1e4047086617d2d28dd7b6

[7] Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/08/beginners-guide-to-convolutional-neural-network-with-implementation-in-python/