Music Genre Recognition

1. Overview

Music is a universal language with versatile varieties. With different characteristics, music has been classified into hundreds of styles, or genres. It has been long for musicians trying to make clear classification and definitions for each music genre, which turns out to be a tough and long-lasting task in the spectrum of music. Nowadays, we can tell that, as the advent of machine learning, classifying music is also intrinsically a classification problem in machine learning. Music Genre Recognition (MRG) is widely used in music apps such as Spotify and AppleMusic not only to label each song with the genre it belongs to, but also to recommend music for their users based on their loved ones.

2. Dataset

Fig.1 GTZAN dataset

In this project, the dataset applied is the famous GTZAN dataset, which is widely used in the area of music style classification, and was first included in the well known paper in music genre classification " Musical genre classification of audio signals " by

G. Tzanetakis and his co-worker P. Cook in IEEE Transactions on Audio and Speech Processing in 2002.

The dataset is made up of 1000 audio tracks each 30 seconds long, which consists of 10 genres, i.e., blues, classical, country, disco, hiphop, jazz, reggae, rock, metal and pop, and with 100 tracks per genre. The format of these tracks are all 22050Hz Mono 16-bit audio files in .wav format.

3. Feature

In Physic, we know that to describe a sound, we have two basic attributes as amplitude and frequency. In terms of a piece of music, or a song, there are also several important properties to describe it, to list a few:

1. Zero Crossing Rate——It is the rate of sign-changes(+/-) along a signal, i.e., the rate at which the signal switches from positive to negative or reverse. Its values are usually higher for songs with strong percussions like metal and rock.

2. Spectral Centroid——It indicates the location of the ”center of mass” for a sound (the weighted mean of the frequencies of the song). Spectral centroid for a blues song will usually locate somewhere near the middle while that for metal would be close to its end.

3. Spectral Rolloff——It represents the frequency below which a specified percentage of the total spectral energy lies.

4. Mel-Frequency Cepstral Coefficients (MFCCs)——a small set of features (usually about 10–20) which concisely describe the overall shape of a spectral envelope. It models the characteristics of the human voice.

5. Chroma Frequencies——the entire spectrum is projected onto 12 bins representing the 12 distinct semitones (or chroma) of the musical octave.

All these properties can be analyzed from a diagram drawn from every piece of music called ‘Spectrogram’.

A spectrogram is a visual representation of the spectrum depicting signals as they vary with time. For sound, its signal for a spectrogram is its frequencies. In 2-dimensional plots, the first axis is frequency while the second axis is time.

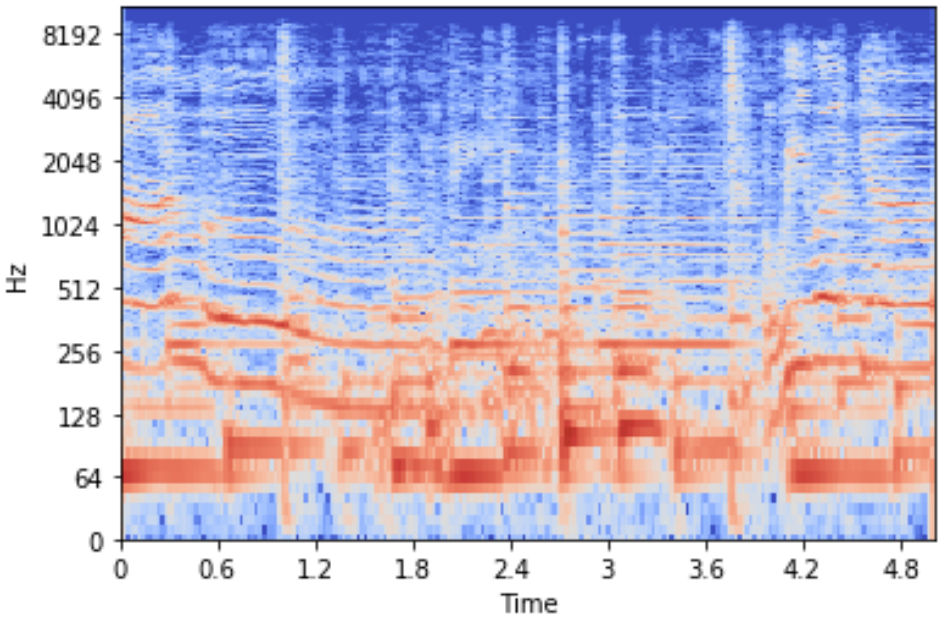
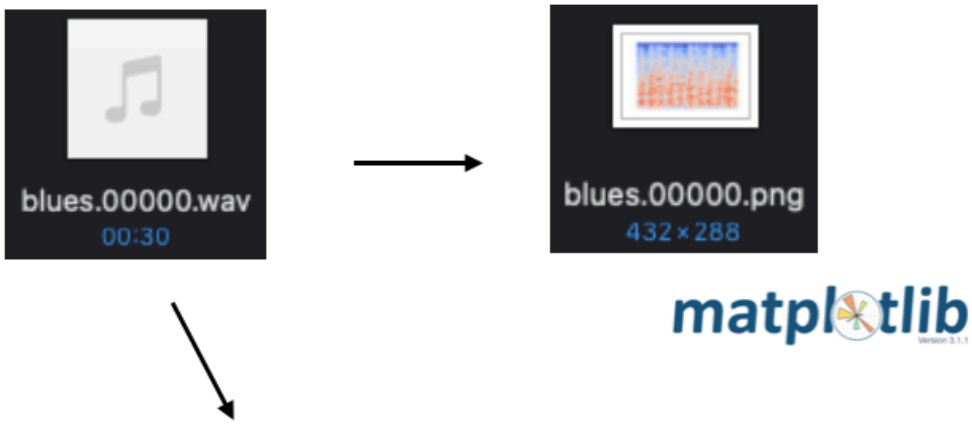


Fig.2 Spectrogram

4. Tool——Librosa

For feature and data extraction, a Python library called Librosa is used, which is a Python module for music and audio analysis, and is more recently used in the field of music.



filename	chroma_stft	rmse	spectral_centroid
blues.00000.wav	0.3420528769493103	0.1244828850030899	1782.62268182674

csv — CSV File Reading and Writing
pathlib — Object-oriented filesystem paths
os — Miscellaneous operating system interfaces

Fig.3 The flow of Librosa

With Librosa, we can extract properties including Root Mean Square (RMS) values and those mentioned above as well as the spectrogram diagram of each audio clip from the original .wav format file.

5. Preprocess

1. Decomposition

Principal Component Analysis (PCA) is a statistical technique to transform a set of possibly correlated variables into a new set of linearly-uncorrelated variables. Those new variables are called Principal Components (PC), whose number is less than or equal to the number of original variables.

PCA is widely used in machine learning for decomposition, i.e., reduce the number of dimensions of a problem. The original set of features is transformed linearly into another set of features, i.e., PCs, which are ranked by decreasing importance.

When training, only one feature PC1 is used as the input instead of the original 26 features, for which shows better results.

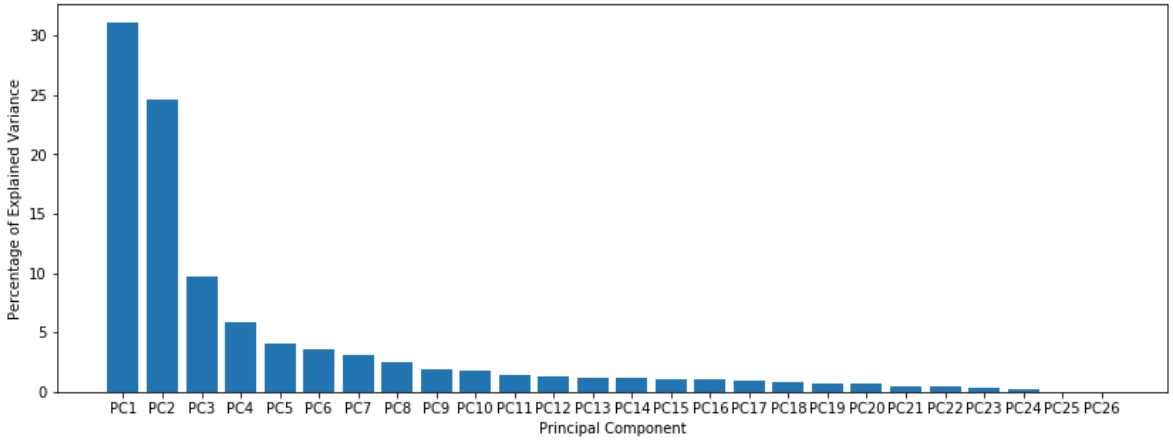


Fig.4 Scree Plot of PCA

2. Dataset split

Total sample (After cleaning): 834

Train: 667

Test = Validation: 167

Using testing data as validation in this project due to the following reason.

During the experiment of using a validation set in hope of reducing overfitting, the results show that without splitting a validation set from the training set results in better accuracy (for most of the time). After digging into the implementation of validation using Keras library, what is found is that the validation set split from the training set by Keras may not update the hyper-parameter, which means that it has no help for improving the training performance. That is to say, the validation set in Keras is only

applied for monitoring the process of training and finding the specific timing of overfitting or determining underfitting.

Also, it is figured out that using the testing set as validation in the Keras context does not cause ‘cheating’, because after simply applying that, the result does not increase dramatically (no increase, to be specific).

6. Model

Several models are attempted through this experiment, including Logistic Regression, Support Vector Machine (SVM), Random Forest, K-Nearest Neighbors (KNN) and Neural Network model. Each shows different performance, which has been demonstrated through the results.

Actually, using a Convolutional Neural Network (CNN) model to train the spectrograms is also attempted, which is a common way for audio analysis in recent years, but its performance shows no better than using the Neural Network model.

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_13 (Dense)             (None, 256)               6912
_________________________________________________________________
dropout_9 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_14 (Dense)             (None, 256)               65792
_________________________________________________________________
dropout_10 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_15 (Dense)             (None, 10)                2570
=================================================================
Total params: 75,274
Trainable params: 75,274
Non-trainable params: 0
_________________________________________________________________
```

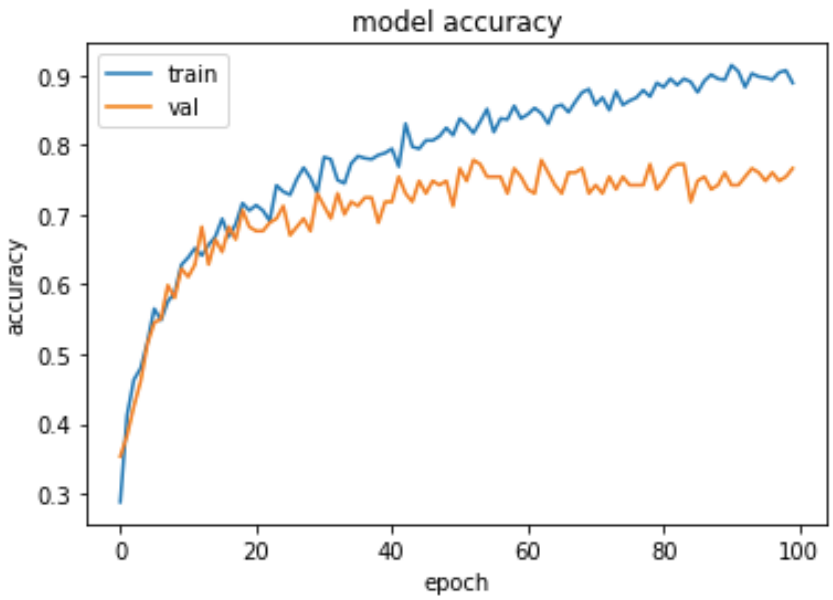
Fig.5 Neural Network model architecture

7. Results and Analysis

As the tables shown below, the Neural Network model demonstrates the highest accuracy. In spite that with different splits with training set and testing set, testing accuracy varies, we can still obtain an average of more than 70% testing accuracy by applying Neural Network model.

Following Neural Network model are SVM, KNN with distance weighting, LogisticRegression and RandomForest.

Model	Accuracy
NeuralNetwork	0.748503
LogisticRegression	0.718563
LibSVM	0.682635
SVM	0.676647
KNN_dis	0.634731
RandomForest	0.610778
KNN	0.610778
DecisionTree	0.550898
Adaboost	0.538922



(b)

Fig.6 (a) Comparison of models (b) Learning curve

From the learning curve above, we can see that after approximately 20th epoch, the model begins to overfit.

8. Discussion and thinking

Compared with the original paper using GTZAN dataset for classification, which results in 61% for nonreal time and 44% for real time. My work, with highest accuracy of 76% and 70+% for most cases using Neural Network model, has improved their results a lot.

However, the performance is still not good enough. No matter what model I use, dense or sparse, simple or complicated, there is always encountered with overfitting problem. Thus, there is still something worth considering with regards to dataset, feature and model:

1. Dirty Data

The reliability of the original dataset is one of the key aspects for the success of machine learning. (The success of machine learning refers to the high performance of the solution by which your problem is solved, e.g. the high accuracy of testing evaluation for a classification problem) If the data in the original dataset is partly broken or dirty, the results of testing accuracy would be definitely affected, or at least unreliable.

In terms of this MGR project, due to the consideration of the nature of diversity of music, as music itself is mixed with different genres and is hard to define by only one genre, the original tracks in the GTZAN dataset are checked. And not surprisingly, it is found that one song from the ‘reggae’ directory is broken, which stops playing from around the 8th second on, some mislabeled songs which are not typical, or at least similar to those in the same category, and several identical songs repeatedly appearing in the same class.

Moreover, data in dataset should be representative, or authentic, as it is what it is labeled.

When different division of the training data and testing data are applied to the same model, the testing results shows severe fluctuation. As a result, the data belonging to one class in the original dataset are likely to be of great variance, or different, to put it simple.

2. Wrong Features

In addition to original dataset that is clean, choosing right features for training is also crucial for the success of machine learning. We should extract most distinctive features from the sample to be classified.

As mentioned, only one feature (PC1) of each sample is used for training, which shows better or no worse results than using any other features, such as using more PCs, e.g., the first 2, the first 5 or the first 10 PCs, and using original features as well. It can be inferred that the original features are not capable of representing the characteristics of each song very well, characteristics that are suitable for classifying the song as the genre it belongs to.

3. Weak Model

Furthermore, efficient and workable models are the core of machine learning.

The poor results and the phenomenon of overfitting may also result from the capability of the selected model. Though a number of models have been chosen for learning, they may all lack the robustness to learn from the features of each given sample so as to predict for new instances instead of just memorizing them.

9. Conclusion and further work

During the presentation, a question and suggestion is raised that why not just using the original audios as the inputs. As far as I'm concerned, all the projects related with audio processing and classification extract useful features, such as MFCCs, from audios for further analysis, rather than just using the original clip. One reason for this might be the data of the original audio is too messy, so extracting useful features that show different properties of the audio reduces the cost of learning greatly and helps machine to learn more efficiently. Moreover, due to the lack of documentation, this project just follows the way how others extract features from audio files, analyze and train with models.

However, just like messy inputs as images are used directly for classification, why can't audios be used as inputs directly? This might be an unsolved task in recent development of machine learning. So the common way now is to transform audio files to image files and to do the familiar image classification. But since audios and images are all saved as a series of values in the computer, there must be an approach to train audios as inputs, and that'll be more direct.

To summarize, this project shows 74.85% testing accuracy of classification for 834 songs belonging to 10 genres using a Neural Network model. Overfitting still exists as the testing accuracy remains much lower than training accuracy while training accuracy is almost 100%. In order to solve overfitting as well as improving performance, more work can be done including adjusting dataset, extracting more efficient features and constructing a more robust model.

Reference

Projects: 1. https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8

2. https://towardsdatascience.com/how-to-apply-machine-learning-and-deep-learning-methods-to-audio-analysis-615e286fcbbc

3. https://gist.github.com/baraldilorenzo/07d7802847aaad0a35d3

4. https://medium.com/x8-the-ai-community/audio-classification-using-cnn-coding-example-f9cbd272269e

5. https://medium.com/@mikesmales/sound-classification-using-deep-learning-8bc2aa1990b7

6. https://github.com/derekahuang/Music-Classification/blob/master/cnn.py

7. Tzanetakis, George & Cook, Perry. (2002). Musical Genre Classification of Audio Signals. IEEE Transactions on Speech and Audio Processing. 10. 293 - 302. 10.1109/TSA.2002.800560.

8. Librosa documentation: https://librosa.github.io/librosa/

9. Dataset: http://marsyas.info/downloads/datasets.html

10. PCA: https://www.youtube.com/watch?v=Lsue2gEM9D0&t=586s

11. Plotting: https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/

12. CNN: https://www.youtube.com/watch?v=LhEMXbjGV_4&list=LLADe5D3g-jb62XZ74pwv9EA&index=22&t=103s

13. Google Colab: https://colab.research.google.com/notebooks/welcome.ipynb