ECE544 Project Report

Mingyu Yang(ymingyu2), Siqi Zhang(siqi5)
University of Illinois at Urbana-Champaign
Department of Electrical and Computer Engineering

1. Introduction

Musical genres are categorical descriptions that are used to describe music. They are commonly used to structure the increasing amounts of music available in digital form on the Web and are important for music information retrieval. A musical genre is characterized by the common characteristics shared by its members. These characteristics typically are related to the instrumentation, rhythmic structure, and harmonic content of the music [1].

Genre is intrinsically related to classification: "assigning" a genre to an item is indeed a useful way of describing what this item shares with other items, of the same genre, and also what makes this item different from items, of other genres. Being able to automatically classify and provide tags to the music present in a user's library, based on genre, would be beneficial for audio streaming services such as Spotify and iTunes. A manual classification of genres can be useful for boot-strapping and evaluating automatic systems, but it is not realistic for large databases, and does not easily scale up [2].

The aim of our project is to implement a classification system, which can determine a music belonging to its most probable class: its "genre", "pop", "electronic", "country", "rap", "jazz", and "light". The network learns the features of a song that makes it more likely or less likely to belong to one genre or another. Then it's able to classify its genre automatically. We use Convolutional Neural Network and Recurrent Neural Network with Keras and Tensorflow to implement this project.

2. Method

2.1. Data Preprocessing

First we convert raw waveforms of music signals to a time-frequency representation and used as input to the system. Time vs Frequency representation of a speech signal is referred to as spectrogram, a 2D representation of a signal, having time on the x-axis and frequency on the y-axis. We have tried several time-frequency representations, including Mel-spectrogram, short-time fourier transform (STFT), and Mel-frequency cepstral coefficients (MFCC), for extracting features, among which Mel-spectrogram proved to be the most effective one.

Initially all the songs are prepocessed as melspectograms. Computing the spectograms makes extensive use of the librosa library [3] for audio processing. We then convert the amplitude spectrogram to dB-scaled spectrogram. The dB-scaled spectrogram is the input of our model. Figure 3 demonstrates the power spectrogram of Mel-spectrogram and STFT.

The mel scale is a perceptual scale of pitches judged by listeners to be equal in distance from one another. The reference point between Mel scale (m) and normal frequency Hertz (f) measurement is defined as

$$m = 2595 \log_{10}(1 + \frac{f}{700}) = 1127 \ln(1 + \frac{f}{700}) \quad (1)$$

$$f = 700(10^{\frac{m}{2595}} - 1) = 700(e^{\frac{m}{1127}} - 1) \quad (2)$$

Seen from the measurements, when f is above about 500 Hz, increasingly large intervals are judged by listeners to produce equal pitch increments. As a result, four octaves on the Hertz scale above 500 Hz are judged to comprise about two octaves on the mel scale. Thus, binning a spectrum into approximately mel frequency spacing widths lets us use spectral information in about the same way as human hearing. Most researchers in audio processing do not use raw spectrograms, which have a linear frequency scale. The mel scale or a logarithmic frequency scale are preferred instead: they reduce the resolution for higher frequencies, which matches the human perception of frequency, and reduce the size of the representation.

Mel Frequency Cepstral Coefficient (MFCC) is a successful feature used in the field of speech processing. A small duration window, for example, 25 milesecond, is considered for processing at a time. This small duration is called a *frame*. To process the whole speech segment by moving the window from beginning to end of the segment consistently with equal steps, called *shift*. Based on the frame-size and frame-shift we get M frames.

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

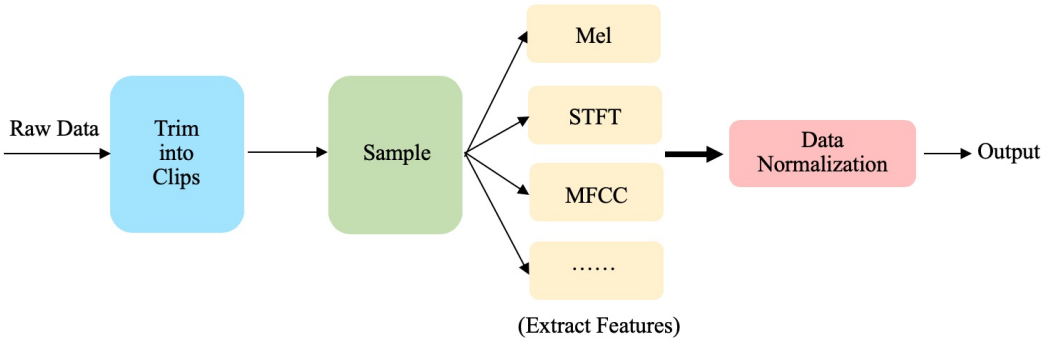

Figure 1. Data Preprocessing

it changes over time. Fourier analysis decomposes a signal into its frequency components and determines their relative strengths. The Fourier transform is defined as

$$F(w) = \int_{-\infty}^{+\infty} f(t)e^{-jwt}dt \tag{3}$$

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(w)e^{jwt}dw \tag{4}$$

In practice, computing STFTs is to divide a longer time signal into shorter segments of equal length and then compute the Fourier transform separately on each shorter segment. This reveals the Fourier spectrum on each shorter segment Then we plot the changing spectra as a function of time. As we often only care about the spectral magnitude and not the phase content in our data processing. The spectrogram shows the intensity of frequencies over time, and spectrogram is simply the squared magnitude of the STFT.

2.2. Model Building

Convert an Mel spectrogram to dB-scaled spectrogram. The dB-scaled spectrogram is the input of our model. Our model uses the spectrogram to reflect the frequency of music data. Then we use a multi-layer 2-D Convolutional Neural Network (CNN) for classification. Since spectrograms are two-dimensional representations of audio frequency spectra over time, attempts have been made in analyzing and processing them with CNNs. It has been shown, that it is possible to process spectrograms as images and perform neural style transfer with CNNs [4]. Then there are 2-D RNNs proceeding the CNN. The model is then called CRNN.

2.2.1 Four-Layer CNN

There are four CNN layers each consisting of five sublayers, 2D convolution, batch normalization, ELU, 2D max-pooling, and dropout layer. Images are 2D signals so 2D convolution is needed here, the mathematical formulation of 2-D convolution is given by

$$y[i,j] = \sum_{m=-\infty}^{+\infty}\sum_{n=-\infty}^{+\infty} h[m,n]\cdot x[i-m, j-n] \tag{5}$$

where, x represents the input image matri x to be convolved with the kernel matrix h to result in a new matrix y, representing the output image. Zero-padding is used in convolution is done in each case where there are no image pixels to overlap the kernel pixels to preserve the image size. So we add Layer 0 consisting of 2-D zero-padding and batch normalization before our 4-layer CNN. Our model draws its strength from making normalization a part of the architecture and performing the normalization for each training mini-batch. Batch Normalization allows us to use much higher learning rates and be less careful about initialization [5]. It also acts as a regularizer, in some cases eliminating the need for Dropout, but we still has a dropout layer for better performance in our case. In contrast to our most familiar ReLUs, ELUs shown as follows

$$f(x) = \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

have negative values which allows them to push mean unit activations closer to zero like batch normalization but with lower computational complexity. Mean shifts toward zero speed up learning by bringing the normal gradient closer to the unit natural gradient because of a reduced bias shift effect [6].

The max-pooling enables CNNs to look at non-overlapping regions of the audio signal and output the maximum value. By eliminating non-maximal values, it reduces computation for upper layers. Also, it provides a form of translation invariance [7]. Dropout consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting. A weak dropout (0.1) is applied between convolutional layers to prevent overfitting of the RNN layers.

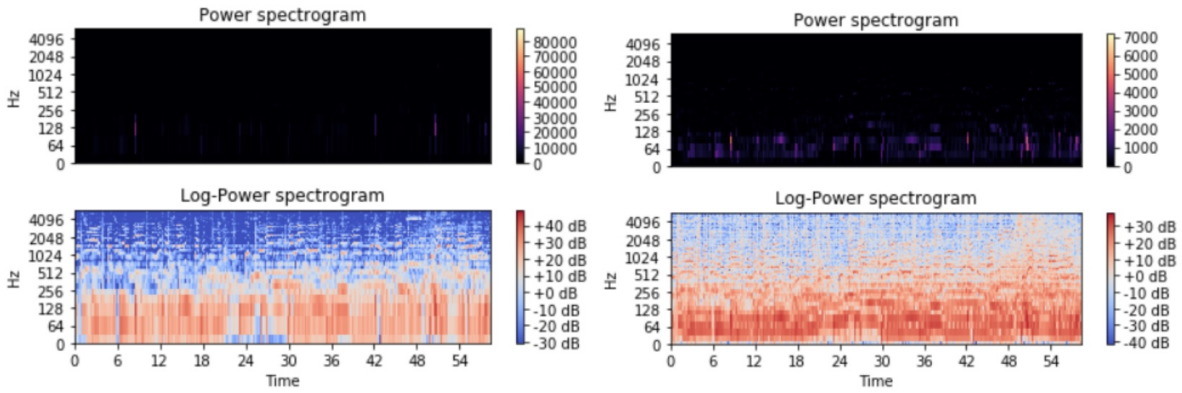

Figure 2. Power Spectrogram of Mel-spectrogram and STFT

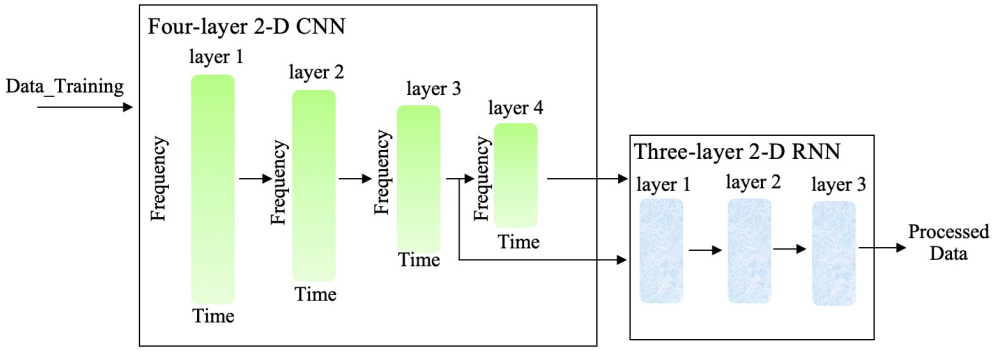

Figure 3. Model Overview

2.2.2 Three-Layer RNN and Dense Layer

CNNs have been combined with recurrent neural networks (RNNs) which are often used to model sequential data such as audio signals or word sequences. RNN are called recurrent because they perform the same function for every single element of a sequence, with the result being dependent on previous computations. Whereas outputs are independent of previous computations in traditional neural networks. Adopting RNN for aggregating the features enables the model to take the global structure into account, rather than only focusing on local features.

There are a type of Recurrent Neural Network that can efficiently learn via gradient descent. Using a gating mechanism, LSTMs are able to recognize and encode long-term patterns. LSTMs are extremely useful to solve problems where the network has to remember information for a long period of time as is the case in music and text generation. The unit consists of three gates: the "forget", "update" and "output" gate. GRU (Gated Recurrent Unit) aims to solve the vanishing gradient problem which comes with a standard recurrent neural network. GRU can also be considered as a variation on the LSTM because both are designed similarly and, in some cases, produce equally excellent results. GRU units have "memory cells" that allow an RNN to capture much longer range dependencies.

A dense layer represents a matrix vector multiplication, and it applies a rotation, scaling, translation transform to change the dimensions of the output vector. In our model, we compare the performance of GRU and LSTM and the results show that GRU performs much better in latency.

2.3. Model Modification and Prediction

We generate the optimized model considering the trade-off between resources and performances by using validation data and training data. And finally, apply this model to predict real music data.

3. Experiments

- Dataset

  Originally we were planning to use the Million Song Dataset [8] and start with its subset of 10k songs. However that dataset doesn't include audio signal, but only metadata of the songs. So we decided to grab songs

of each genres by ourselves from music platforms. We manually downloaded 1800 songs from QQ music application. There are a large number of songs classified by humans in the music application and categorized into genres and subgenres. We downloaded these marked songs in batches as our dataset. There are six genres in the dataset, with 300 songs in each genre. The format of the data is mp3 and the bit rate is 128 kbps. We randomly selected 60%, 1080 songs, as the training set, 20% as the validation set, and 20% as the test set from the 1800 songs.

- Pre-Processing

  Firstly, with librosa [9], we trim each song into a 25 seconds fragment which is in the center of this song and then downsample them to 12kHz. We choose hop length, the number of samples between successive frames, as 256, and the length of the FFT window as 512, and Mel-frequency spectrogram bins as 96. Then, we generate the Mel-gram of each song and convert it to dB-scaled spectrogram, whose size is 96 * 1172, which is the input of our CRNN model. The input data of our CNN will be vectors of time and frequency along with tags. As shown in Table 1. The input data of our RNN, as shown in Table 2 (for example, set look_back = 2).

- CRNN Parameters

  For the CRNN model, we use Keras with Tensorflow backend to implement. We use ADAM for optimizer function and categorical cross entropy function for loss function. Both the dropout layer rate and RNN dropout rate are set to be 0.1. We use batch optimization for accelerating the training process, with batch size equal to 16. And we use loss value and accuracy for model evaluation.

- Results

  After building the model, we use the validation dataset to modify the model. The validation data is treated totally the same as previous training data after being applied to the neural network. First, we try to run the model on our PCs, but the operation time is too long. So we deploy our model on Google Cloud Platform. We used GPU enabled servers to accelerate the training process. With the help of 20 GPUs, the training time of each epoch can reach 30 s. After repeatedly model modifications with training data set and validation data set, we use the best model to make predictions on testing data set. After 200 epochs, the accuracy reaches in 83.42% the training set, in validation set 81.28% as shown in Figure 4 and 5.

4. Discussion and Conclusion

There's some existing work on using neural networks for music genre classification, and most of them use CNN after obtaining spectrum data and then the job is turned into image processing. Several projects on Github have implement music auto-tagging functions, like "MSD_split_for_tagging" [10], "music-auto-tagging-keras" [11], "DeepAudioClassfication" [12]. Also, several papers also discuss different methods and thoughts for this issue, for example, Automatic tagging using deep convolutional neural networks [13], Music emotion classification and context-based music recommendation [14], Classification of Musical Genre: A Machine Learning Approach [15]. A popular article on recommending music at Spotify shows the use of convolutional neural networks for music genre classification. This article uses an architecture that consists of 3 convolutional plus max pooling layers and finally max pooling, average pooling and L2 pooling concatenated and fed into three fully connected layers. The article uses melspectograms for song preprocessing, which seems to be the standard approach for song clips and as of now, giving the best results [16].

The models are effective and efficient, but still, have drawbacks. The first is whether the choosen time-frequency representation is the best in certain problem settings. The other is that although CNNs are fast, the accuracy drops greatly when applying batch normalization. Based on the problems we see, we choose to insert RNNs to our model, which brings cost in latency. However, the cost of time is tolerable because we use GRU which brings much better accuracy and is efficient with low latency. We choose the parameters in the max pooling layer between CNN and successive RNN carefully to reduce the number of parameters and thus making the data fit into RNNs perfectly.

- Problems

  Latency is still a major problem because although CNNs are efficient when processing large dataset, RNNs performs poorly in runtime because it's good at processing smaller dataset. Our focus should be how to effectively accelerate the model, especially when there are 3 layer RNNs. In our training process, we set a larger batch size because the latency of each epoch increases, thus resulting in lower accuracy. All the parameters in the present model have been optimized, but the optimal is yet to come. As mentioned before, we will try to improve our accuracy and efficiency and find the optimal trade-off point. There are many aspects to consider, experiment on more, and make possible optimizations, sampling rate, hop length, mel bins, data trims of the songs, resource, a trade-off between speed and memory, parameters, layer dimensions, and batch size.

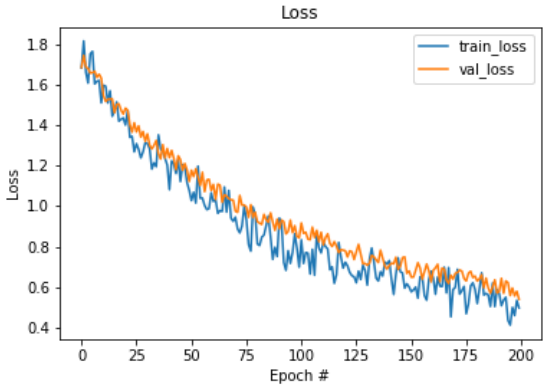

Figure 4. Loss of training and validation data

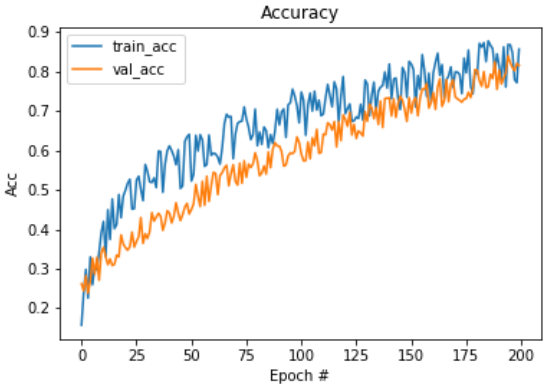

Figure 5. Accuracy of training and validation data

Table 1. Input of CNN.

X	Y
x1[[time vector], [frequency vector]]	1 (Stands for Pop)
x2[[time vector], [frequency vector]]	5 (Stands for Rock)
...	...

Table 2. Input of RNN.

X	Y
x1[(t-1, [frequency(t-1)]), (t, [frequency(t-2)])]	1 (Stands for Pop)
x1[(t-1, [frequency(t-1)]], (t-2, [frequency(t-2)]]]	5 (Stands for Rock)
...	...

- Future Work

  We only has six genres in the classification framework, and are looking forward to expanding the scale to more than 10 genres. Also, there are many tags/characteristics of songs except genres, for example, mood conveyed, the instrument played, era created. We can achieve a thorough and accurate music classification if all the features are concerned. For example, telling who the singer is when the device detects a song and music recommendation system are all potential application field.

References

[1] George Tzanetakis, and Perry Cook. *Musical genre classification of audio signals*. IEEE Transactions on speech and audio processing, 10(5):293–302, 2002.

[2] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. *Representing Musical Genre: A State of the Art*. In Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011, pages 591– 596, 2011.

[3] B. McFee, M. McVicar, O. Nieto, S. Balke, C. Thome, D. Liang, E. Battenberg, J. Moore, R. Bittner, R. Yamamoto, D. Ellis, F.-R. Stoter, D. Repetto, S. Waloschek, C. Carr, S. Kranzler, K. Choi, P. Viktorin, J. F. Santos, A. Holovaty, W. Pimenta, and H. Lee. *librosa 0.5.0*. 2017.

[4] Prateek Verma, and Julius O. Smith. *Neural Style Transfer for Audio Spectrograms*. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 2002.

[5] Sergey Ioffe, and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. arXiv preprint arXiv:1502.03167, 2015.

[6] Djork-Arne Clevert, Thomas Unterthiner, and Sepp Hochreiter. *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. arXiv preprint arXiv:1511.07289, 2015.

[7] Weibin Zhang,Wenkang Lei, Xiangmin Xu, and Xiaofeng Xing *Improved Music Genre Classification with Convolutional Neural Networks*. INTERSPEECH 2016, San Francisco, USA, September 8–12, 2016

[8] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. *MSD: The million song dataset*. In Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011, pages 591–596, 2011.

[9] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. *librosa: Audio and music signal analysis in python*. in Proceedings of the 14th Python in Science Conference, 2015.

[10] "MSD_split_for_tagging", *https://github.com/keunwoochoi/MSD_split_for_tagging/*

[11] "Music-auto-tagging-keras", *https://github.com/keunwoochoi/music-auto_tagging-keras.git*

[12] "DeepAudioClassfication", *https://github.com/despoisj/DeepAudioClassification*

[13] Keunwoo Choi, Gyorgy Fazekas, and Mark Sandler. *Automatic tagging using deep convolutional neural networks*. 17th International Society for Music Information Retrieval Conference, 2016.

[14] Byeong-jun Han, Seungmin Rho, Sanghoon Jun, and Eenjun Hwang. *Music emotion classification and context-based music recommendation*. Multimedia Tools and Applications, May 2010, Volume 47, Issue 3, pp 433–460.

[15] Roberto Basili, Alfredo Serafini, and Armando Stellato. *CLASSIFICATION OF MUSICAL GENRE: A MACHINE LEARNING APPROACH*. Published 2004 in ISMIR.

[16] S. Dieleman "Recommending music on Spotify.", *http://benanne.github.io/2014/08/05/spotify-cnns.html* 2014. [Online; accessed 15-Nov 2018].