Introduction

Audio data is becoming an important part of machine learning. We are using audio to interact with smart agents like Siri and Alexa. Audio processing, classifying, recommending also lies at the heath of music provider platforms, such as Spotify, iTunes as well as music discovery engines including Shazam. One of the current problems about audio is that how it should be represented. The raw input could be a *.wav* file, containing the pure signal in which are no metadata on instruments included, author, etc. The pure signal isn't very handy though - mainly because it's quite heavy. Feeding it to a pure LSTM is an option, but since it's computationally expensive, it not the most efficient approach.

The rhythmic patterns, chords, chord progress, melodic lines, tonality of musics can be utilized to describe and differentiate audio signals as elaborated features in order to identify the various genres. However, the extraction of these features from an audio recording along with the implementation of a classification algorithm using those are difficult in general and they require an expert's knowledge in musics[1]. So, one can exploit the frequency domain representation of audio recordings to classify various genres. This can be simply performed via an FFT algorithm to extract feature maps and then by taking the advantage of the computational power of the contemporary computers, a deep learning model can be trained with these representations and used as a classifier.

In deep learning applications, preparation of dataset is one of the most crucial parts of the effectiveness of a method. If a dataset is large enough and is appropriate for the at hand, then a deep learning model will gain the ability of categorizing input samples faithfully. We used a portion of Free Music Archive (FMA) [2][3] to train and test our deep learning model. The subset that we utilized consists of 8000 songs with 8 genres including Electronic, Experimental, Folk, Hip-Hop, Instrumental, International, Pop and Rock.

Methodology

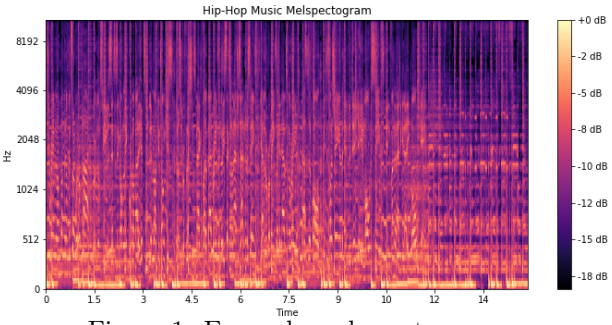

Figure 1: Example mel-spectogram

The implementation steps are explained as follows:
First, the mel-spectograms of 30 seconds clips are generated from *.wav* files. A regular spectogram is a visual representation of the frequencies over time. It is the squared magnitude of the short time Fourier transform (STFT) of the audio recordings. STFT of an audio signal is computed as a succession of FFTs of windowed data frames of length M, where the window slides forward through time. The window hops over the original signal at intervals of R samples. So, the 2 parameters, M, and R, have a vital effect on the generated spectogram of audio recordings. We found out that the shortest reasonable period a human ear can distinguish is about $100ms$[4]; therefore, we used window length, M, as 2048 samples which resulted in about $92ms$ time resolution for a song in the dataset where the sample rate of the data is $22050Hz$, and number of FFTs and the hop length, R, are chosen as 2048 and 1024, respectively. So, the spectogram will have 2 dimensions where $x-axis$ represents time and consists of 646 number of points ($ceil$(duration $\times$ sampling rate/hop length)), and $y-axis$ represents frequency with 128 points. A mel-spectogram is the mel-scaled version of a regular spectogram which approximates human auditory system's response more closely than linearly-spaced frequencies. An example of mel-spectogram is shown in the Figure 1.

After obtaining the mel-spectograms of each audio recordings, the 80% of spectograms are used to train our deep learning model. The remaining 10% of the spectograms is used for validation and the other 10% is used for testing. The deep learning model is given in the Figure 2.

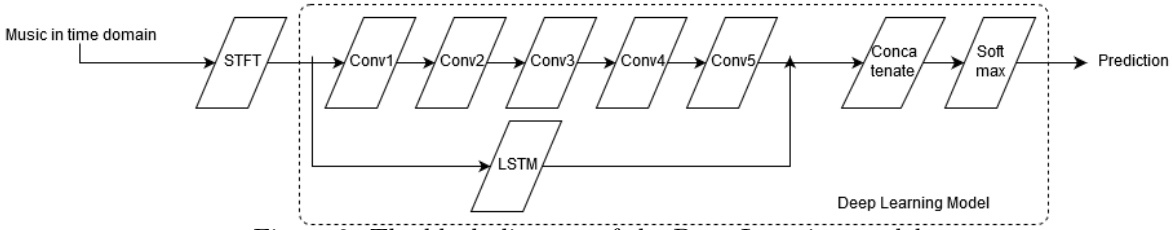

Figure 2: The block diagram of the Deep Learning model

Our deep learning architecture has 2 stack of layers, one of which is composed of convolutional neural networks (CNNs) followed by rectified-linear units (ReLU) as activation function followed by with 2×2 and 4×4 max-pooling layers using 2 strides. The bottom branch has 4×2 max-pooling followed by long short-term memory (LSTM). The output of these two parallel paths are concatenated and passed through last fully connected (FC) layer. Finally,

predictions are obtained as the output of soft-max (SM) layer. Since we want to preserve differentiating power of frequency spectrum, we set the size of CNNs kernels as 3 *width* and 1 *height* to make use of correlation in time. LSTM, however, has the ability to find both dependencies across short period of time, and a long term structure of a song. By combining the complementary features of these layers, we aimed to achieve better classification performance. Finally, we selected cross entropy function as the loss function as trained for 50 epochs. A in detail illustration of our model is given in presentation slides.

Results

The loss values and accuracy of the model during training and testing in terms of a function of epoch as well as the confusion matrix of the test spectrograms are shown in the Figure 3, respectively.

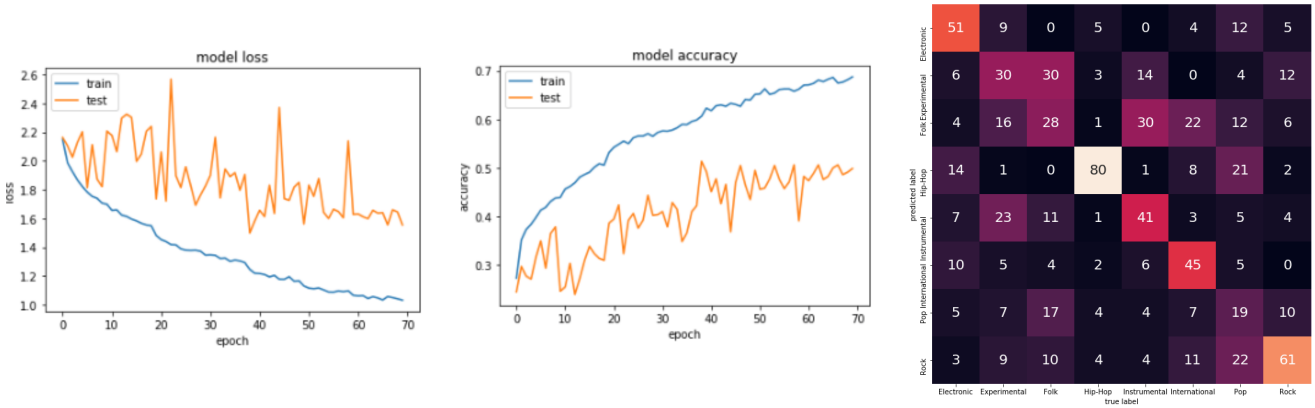

Figure 3: The loss values and accuracy of the model in terms of epoch, and the confusion matrix of the test specs.

So, it can be seen from the loss values and the accuracy scores of the model that the performance of the model increases proportionally with the number of epochs. Actually, the model is trained with different parameters of mel-spectogram but the one which gives the best result is mentioned as the parameters in the introduction part. The ordering of the genres in the confusion matrix is the same as in introduction. It can be observed that the trained model can successfully identify the Hip-Hop musics whereas it performs comparable poor when it comes to distinguishing Pop clips from other genres. All in all, using spectograms with such a small network yielded satisfactory results.

Conclusion

The frequency representation of an audio recording can be used to categorize since the same genres have similar frequency spectrum. The frequency features of audio recordings are extracted by calculating their mel-spectograms. Then, these spectograms are fed to our deep neural network train the model and test its performance. The model is composed of both CNN and RNN layers in parallel form and exploits not only the correlation of frequency features in time (CNN) but also long dependencies in a given music spectogram (RNN).

It is proved to be difficult to find and manipulate a suitable dataset for classifying music clips. Our starting was to identify composers of classical musics; however, we performed extensive dataset search and were able to found 1-2 suitable corpus for our training purposes but those ones also didn't include enough musics for each composer. Furthermore, the discrimination ability of spectograms in terms of composer seemed to be not sufficient since the clips from the same genre highly alike such that a simple network architecture was not able to catch those little nuances. Hence, all of the network models were unsuccessful to differentiate composers even though we try our many hyper-parameters of spectograms. All in all, we decided that composer identification is a substantially hard problem and move on to genre classification problem for which available dataset are more abundant.

References

[1] Gianluca Micchi. A neural network for composer classification. International Society for Music Information Retrieval Conference (ISMIR 2018), 2018, Paris, France. (hal-01879276)

[2] Defferrard, M., Benzi, K., Vandergheynst, P., & Bresson, X. (2016). FMA: A Dataset for Music Analysis. ArXiv, abs/1612.01840.

[3] Free Music Archive.[online] Available at: https://freemusicarchive.org/ [Accessed 3 Jan. 2020].

[4] Michalak, P. (2019). Music Genre Recognition. [online] DeepSound. Available at: http://deepsound.io/music_genre_recognition.html [Accessed 3 Jan. 2020].