

24th April 2022

Music Genre Classification

50.039 - Theory and Practice of Deep Learning

Submitted by:

Jodi Tan - 1004587

Priscilla Tan - 1004668

Wang Zixuan - 1004448

Shwetha Selvam - 1004637

Theint Myat Wai - 1004358

Table of Content

Table of Content	2
Introduction	3
Data	3
Data Preprocessing	4
CNN	4
LSTM	4
Methodology	5
Basic CNN	5
LSTM	5
Model Performance Results	7
Github Repo Link	7
Results Discussion	7
Basic CNN results:	7
Simple LSTM Results:	10
Comparison With State-of-The-Art Models	11
References:	11

Introduction

Target Audience: Anyone who would like to discover a new genre of music or similar songs to ones they enjoy.

Problem: Music Genre Classification

We are creating a model that receives a clip from a song, and produces a predicted class corresponding to the music genre of the sound sample, according to the different models.

This model has many possible applications. For example, you hear of a song at a mall that you enjoy. And after using shazam to identify the title of the song, you know the title of the song, but cannot personally identify why you like the song and how to find other songs similar to it. Our model will be able to pick up on key elements of the song which would help its users by recommending a newfound genre of music to enjoy!

Data

These are the datasets we considered, with a brief summary of their characteristics:

- a. [GTZAN dataset](#) (the MNIST of sounds)
 - Collected in 2000-2001
 - 30 second audio clip
 - 10 genres with 100 clips each

- b. [FreeMusicArchive](#) (repository of audio segments with relevant labels and metadata)
 - Collected in 2017
 - 30 seconds in length
 - contains 8,000 audio segments
 - 8 genres

We used the GTZAN dataset for the actual dataset as it had only 10 output genres while FreeMusicArchive had 161 genres. Using libraries such as librosa and matplotlib, we cleaned the data into 6 folders to be used for training and testing. The original 30 second clip for Data/genres_original/jazz/jazz.00054.wav could not be processed due to incompatibility with the libraries, hence we did not use this file.

Data Cleaning: Outputting Different Audio Images into Folders

1. Wavelets
2. Spectrogram

3. Melspectrogram (spectrogram with mel scaling, which works better according to our research)
4. Mfcc (works better for human speech)

Data Augmentation of Melspect folder: 2 methods

5. SplitmelSpect (split each audio file into 3 parts, and create 3 Mel-spectrogram from the original 1 audio clip)
6. melspectMask (using a custom function to mask the Mel-spectrogram using rectangles in the x and y axis, so each original audio clip produces 5 variations of spectrograms)

Data Preprocessing

CNN

We used images from different folders to train the CNN model.

LSTM

The GTZAN dataset is also used for the LSTM model. Since LSTM takes an input of the sequence length, batch size and features instead of an image (CNN input), our approach is to extract mfcc, chroma and spectral centre features from the GTZAN features_30_seconds csv file which will then be fed to the LSTM model.

Spectrograms and mel spectrograms are images in their basic form. Which would mean that typical CNN operations such as convolution and pooling can be done. However for LSTM we have only utilised the characteristics highlighted above. Due to our unfamiliarity with such foreign data we were not sure of how to proceed on with data manipulation. We had the choice to either use the features as it is or to change and potentially tamper with it. Since we did not want to risk tampering with the data of mfcc, chroma and spectral centre as we did not carry out any manipulation on the dataset.

Because of the time constraint and main focus on LSTM, we later used a pre-processed data from a github source¹ which uses a similar approach as our planned project approach, in an extensive way. Because of this limitation, the data for the LSTM model is only 8 music genres: Classical, Country, Disco, Hiphop, Jazz, Metal, Pop and Reggae.

¹ <https://github.com/ruohoruotsi/LSTM-Music-Genre-Classification>

Methodology

In this project, CNN (Convolutional Neural Network) model and Simple LSTM (Long Short-Term Memory) are mainly used to predict the music genre from a song file, either in .au format or .wav format.

Basic CNN

After preprocessing the data, we create our first model. We construct a Convolution Neural Network model with required input and out units. The final architecture of our CNN model is shown in the figure below. We used 3 types of data (spectrogram, wavelet and splitmelspec) for the training and testing, with our results shown in the next section. Our CNN Model contains 2 convolutional layers, the first one outputting 6 of 3 by 3 filters and the second outputting 16 of 3 by 3 filters. We also apply max pooling with size 2 by 2. After flattening, we put this into 3 different fully connected layers. We tried adding dropout regularisation into the models with 0.2 and 0.4 probability of the neuron being deactivated as comparison. We also experimented with large and small learning rates to see how it will impact the training and validation accuracy and the losses.

Layer (type:depth-idx)	Param #
ConvolutionalNetwork	--
└─Conv2d: 1-1	456
└─Conv2d: 1-2	880
└─Linear: 1-3	14,246,520
└─Linear: 1-4	10,164
└─Linear: 1-5	850
Total params: 14,258,870	
Trainable params: 14,258,870	
Non-trainable params: 0	

LSTM

In addition to the CNN model, the Long Short Term Memory LSTM model is also created to compare the performance with the CNN model. A LSTM neural network model is constructed with the inputs, hidden layers and output units. We experimented with 3 different hidden dimensions: 128, 256, and 512 for unidirectional, and 2-layer LSTM to achieve the plausible results. The final architecture of our LSTM model is shown in the figure below with a hidden dimension of 128. MFCC, chroma and spectral centre features of the songs (33 in total) are fed to the LSTM model along with the batch_size and the sequence length. Sequence length is obtained by dividing the timeseries data of the song by the hop length. After

the LSTM layer, the data is passed through the fully-connected layer with an output dimension of 8, since the total number of music genres for this model is 8. Finally, in the forward function, a softmax function is used to predict the music genre of the input.

Layer (type:depth-idx)	Param #
LSTM	--
└─LSTM: 1-1	215,552
└─Linear: 1-2	1,032
Total params: 216,584	
Trainable params: 216,584	
Non-trainable params: 0	

One limitation of our model is using the low number of epochs when training the model. Though we know that a higher number of epochs such as 500 is required to get a model with higher accuracy, our experimentations were done with the numbers of epochs ranging from 10 to 50 only because of the lack of GPU which can speed up training our model. Initially we had 30 epochs and that resulted in a subpar performance as the loss was too high but adjusting it to 40 led to a better performance overall. However, the accuracy and losses data have multiple fluctuations hence we increased the epoch to 50, in which the model performance becomes higher with less fluctuation. With a GPU/CUDA, our model has potential to give more accurate predictions.

Afterall, our LSTM model is able to predict 8 music genres with an input of .au music file for an accuracy of 67.14%.

If time permits, we aim to address the shortcomings in our model. For example, we would have explored how to use spectrograms and mel spectrograms as an input to our model to derive our results. Since it seems to be a standard way to carry out deep learning for RNNs we can probably look into it for more reliable results. But more importantly, our focus would be directed towards our model being able to classify all 10 songs into their genres. Currently, our model can only do so for 8 classes.

Model Performance Results

Github Repo Link

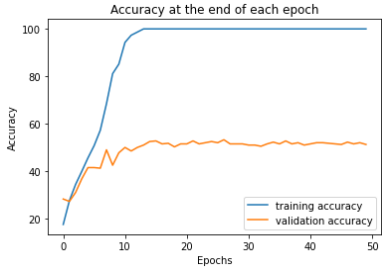
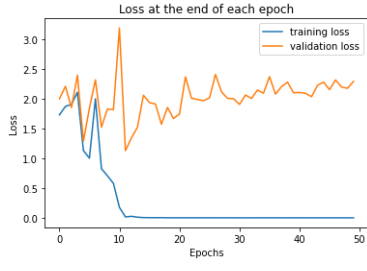
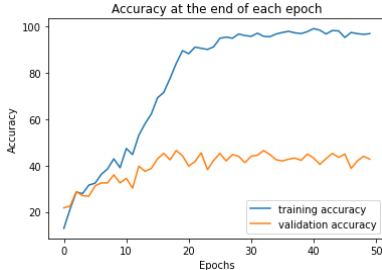
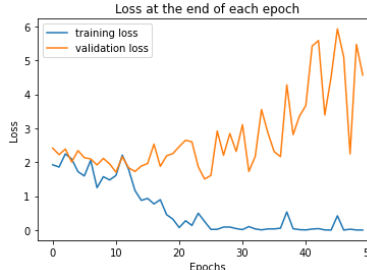
<https://github.com/prispearly/music-genre-classification>

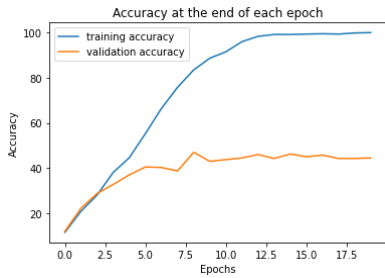
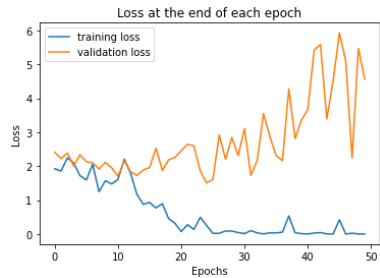
Basic CNN results:

In order to fine tune our hyperparamters, we have tested the data for that of with/without dropout regularisation with different probability of deactivating neurons values at 0.2 and 0.4, with varying number of fully activated layers, and also a low and a high learning rate of 0.001 compared to 0.1. We have decided to use the Adam optimiser algorithm. When we first ran 50 epochs of the first model, as we can see from the results, we realised that the accuracy becomes stagnant after 20 epochs, so we can reduce our epoch size for the next few iterations of the model to running only 20 epochs.

When we use a high learning rate, such as 0.1, the accuracy will converge to a lower value quickly and across the number of epochs, the accuracy will not increase. As such, we decided that the best value for the learning rate in our case would be 0.001.

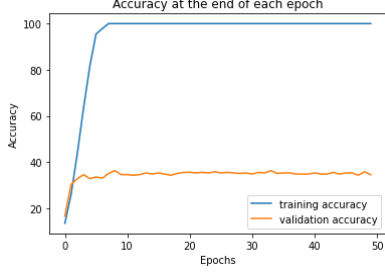

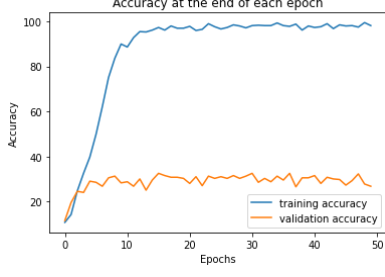
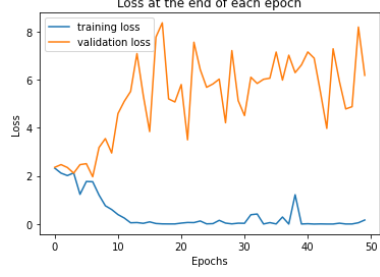
1. Spectrogram Image Folder

Details of model	Epochs	Dropout		
3FC layer Learning rate=0.001	50	0		
3FC layer Learning rate=0.001 Adam Optimiser Algorithm	50	0.2		

2 FC layer	20	0.2		
------------	----	-----	---	---

The highest validation accuracy at the end of an epoch achieved in this model would be 47%. The model will more or less hover around 45% maximum validation accuracy. The accuracy for different parameters stays more or less the same, whether there is dropout or not or whether there is a change in number of fully connected layers or not. Without the dropout, the values are more consistent and less fluctuation is seen in the model results.

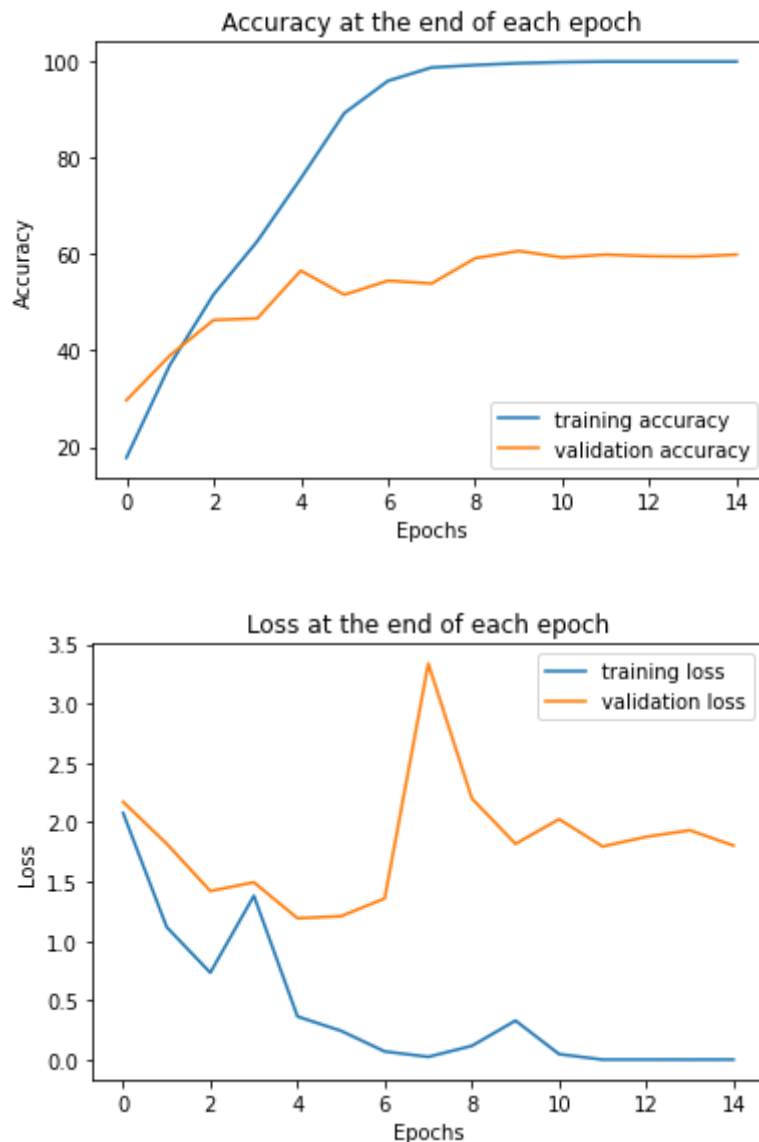
2. Wavelet Image Folder

Details of model	Epochs	Dropout		
3FC layer Learning rate=0.001	50	0		
3FC layer Learning rate=0.001 Adam Optimiser Algorithm	50	0.2		

The highest validation accuracy observed is only 32.5%. As such, we would not be using wavelet data to train our Basic CNN model.

3. Splitmelspec Image Folder

Using 15 Epochs, learning rate of 0.001 and no dropout:



The highest validation accuracy that we got from this model would be 60.7%.

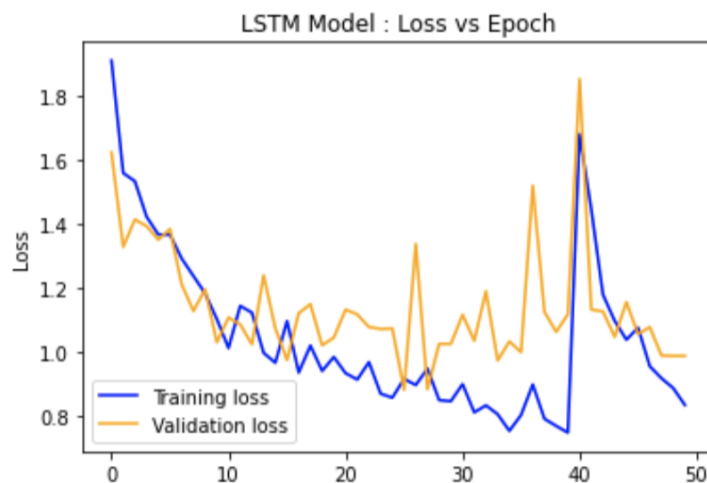
In conclusion, we have narrowed down our best CNN model parameters that gives us the best validation accuracy to:

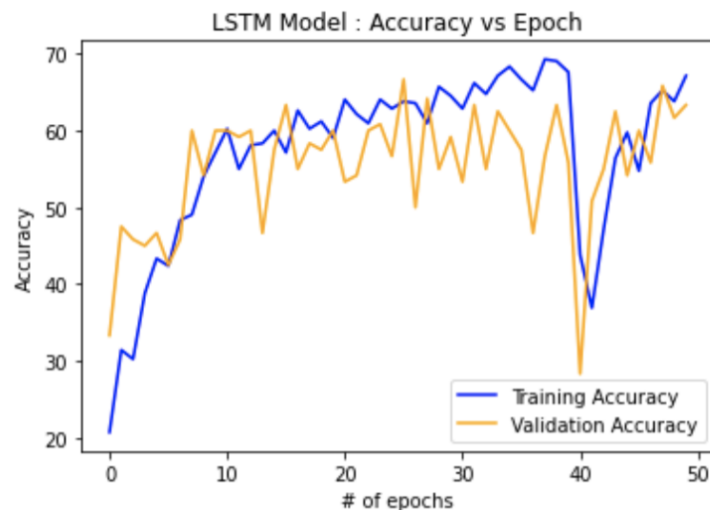
- learning rate of 0.001
- using the Adam Optimiser(Adaptive Moment Estimation) algorithm,
- without dropout
- Training with the splitmelspec image folder
- 3 FC Layer

The best model for basic CNN will be found under BasicCNN_splitmelspec.ipynb in the repository.

Simple LSTM Results:

We have tested the LSTM model with different numbers of LSTM layers : 2, and 4, while changing the hidden dimension to 64, 256, 512 from 128. We observed that having 4 LSTM layers stacked on each other gives an underfitting result. Though we note that changing to higher hidden dimensions (256, and 512) gives higher accuracy, it takes longer time and had to stop manually because of the time constraints. A hidden dimension of 64 with 4 LSTM layers also did not have a plausible result. When we ran with 20 epochs, the accuracy was only 48%, and increasing it to 30 epochs also gave fluctuating results. From our experiments, the model with 2 LSTM layers and a hidden dimension of 128 trained for 50 epoches gives the optimal result with 67.14% and the performance can be seen in the figures below.





Comparison With State-of-The-Art Models

Although thorough research was done on the internet, we were not able to find out any cutting edge models being used commercially. Hence, we had to resort to research and thesis papers for legitimate and reputed models. One such model was the advanced CNN model used in the paper of “Music Genre Classification”(Huang et al., 2018). In their findings it has been reported that 82% accuracy was achieved as compared to other efficient models such as SVM, K means clustering and a regular neural network when an advanced CNN was used. The CNN used was 4 x 4 and had 3 convolutional and alternating max-pooling layers along with it. On top of that each fully-connected layer at the end had a RELU activation function applied to it. In the end it was passed through a softmax layer to classify each input with its genre. The model was refined through the usage of the cross-entropy loss regularisation technique. This goes to show that using this combination is more likely to produce better and more reliable results. Perhaps our model could replicate some of the features listed over here to improvise the performance in the future.

Despite it producing better scores than our model there are some similarities making our model advanced too. They are the usage of mel-spectrograms as inputs rather than just audio files. It has been evidenced in the paper that these audio files caused overfitting. Additionally, audio files indicate the loudness of an audio sample rather than extracting all of its features making the data skewed.

All in all, it can be deduced that even though our project into music genre classification may have a few flaws due to constraints it still functions decently and is similar to industry level standards.

References:

Github, Sept 4, 2021

<https://github.com/ruohoruotsi/LSTM-Music-Genre-Classification>

Github, June 14, 2021

https://github.com/sawan16/Genre-Classification-using-Deep-learning/blob/main/genre_classification.ipynb

Huang, Serafini, Pugh. 2018. Music Genre Classification.

<http://cs229.stanford.edu/proj2018/report/21.pdf>