Music Genre Classification Using Neural Networks

Shiraz Yaacob (ID. 312483456), Oran Goldrich (ID. 205881162)

Submitted as final project report for Deep Learning, IDC, 2020

1 Introduction

During the semester we worked together on our assignments. Every meeting we knew exactly what music we are going to listen to while coding- lo-fi hip hop music. When we needed to choose subject to our project, we knew it should be something we are both passionate about- music.
As humans, detecting song genres is a rather simple task, but not for a machine. With music being released on a daily basis over global platforms, the need for accurate songs criterion is rising rapidly. An automatic, simple way to achieve such classification is necessary functionality for many companies: Spotify, Apple Music, SoundCloud etc.
With the increasing use of Machine Learning and Deep Learning, we decided to use Deep Learning to classify music genres. We implemented a few types of Neural Networks and tested it over data-sets of classified songs, pre-processing them first into mel-spectrograms. We focused on different architectures of Neural Networks- CNN, RNN, and CRNN. After training over more than 6000 instances, we tested over validation and plotted the accuracy and loss. We examined the graphs and discussed insights.
We found out that song classification into genre is a rather complicated task when the instances are mel-spectrograms. While different architectures of CNN provided satisfying results, RNN functioned poorly. CRNNs achieved almost the same result as CNNs.

1.1 Related Works

Music genre classification is not a new problem in Machine Learning. Many have tried to implement different algorithms to tackle it.
As a benchmark, human accuracy is around 70% for genre classification [3]. Some works, such as [1], classified music genres according to tempo (BPM - Beats per Minute) using SVM. This work shows decent results with 67% accuracy over 8 genres. In recent years, using MFCC (Mel-frequency cepstral coefficients) is one of the leading approaches for music genre classification [2][4]. Top-notch models today achieved 91% accuracy over 10 genres [4].

2 Solution

2.1 Pre-Processing

While searching for a data-set we encountered the use of mel-spectrogram as instances, instead of sound files (such as mp3). It is a wildly used technique to pre-process audio data before applying methods for sound processing, especially Music Information Retrieval (MIR) [5].

A mel-spectrogram is a representation of sound.
By calculating the short-term distribution of power into frequency components (wavelength), we get the intensity of each frequency at a given time-window (frame). Stacking all the frames together produces mel-spectrograms.

As a data-set, we decided to use FMA- 8 GB of raw audio files [6], pre-processed by Priya Dwivedi [7]. The FMA, Free Music Archive, is an open data-set suitable for evaluating several tasks in MIR, a field concerned with browsing, searching, and organizing large music collections. The FMA includes 917 GiB (343 days) of audio from over 100,000 tracks. The audio comes from more than 16,000 artists, and each song was cut to samples of 30 seconds. The FMA provides three subset of the full data- small, medium, and large, which include 8K, 25K and 106K of trimmed songs respectively. Since we can't process such large scale data-sets, we choose to use the small one.

Before creating mel-spectrograms, some parameters should be decided. The most important parameter used in the transformation is window length. This parameter indicates the frame (window of time) which goes through Fourier transform. Another parameter is hop length- the number of samples between successive frames.

The shortest reasonable period of time a human ear can distinguish is about 10ms, so this is the typical window length for this transformation. The hop length is 512, so the mel-spectrograms produced by librosa has a shape of 640 * 128 [8]. The frame placed at X axis, the frequency at Y axis, and each cell color describes loudness in decibels (dB).

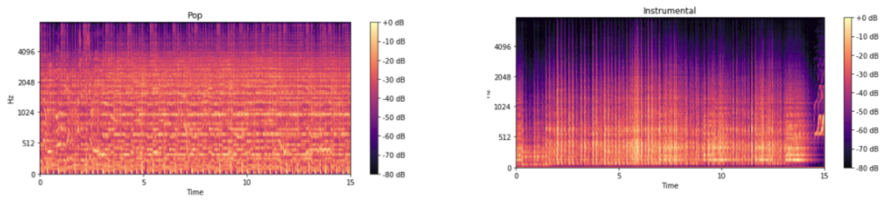

Figure 1: The difference between mel-spectrograms from two genres - Pop and Instrumental.

Viewing the mel-spectrograms of different genres, it is clear that we can classify songs using mel-spectrograms as an input. As shown in Figure 1, the difference between each genre is obvious.

2.2 General approach

Our general approach to this classification problem has a few steps. In the first step, as we treat a spectrogram of a song as an image, we implemented different CNNs. Later on, as our data embodies a time dimension, we put a RNN into use. RNNs might be able to aid identifying the short and long term temporal features in the song. The last step was to combine the two into a CRNN attempting to take advantage of local feature extraction of CNNs with temporal summarization of a RNN. In all our models we used soft-max classifier. Also, we used L2 regularization to reduce over-fit.

Firstly, we implemented two 2D CNN models - A shallow and a deeper one. Our main line of thought focused on the fact that the complexity of our images is low. Unlike usual images, mel-spectrograms contain rather simple shapes. Following this line, we hypothesize that a shallow CNN model might have similar or better performance compared to a deeper model. After some research, it seems most papers do not implement deep convolutional networks for this task, but we want to make sure this is just. Our shallow model is 4-layer 2D CNN + 3 fully-connected. The deeper model is a 10-layer 2D CNN + 3 Fully-Connected model. We used batch normalization and max-pooling. In the deeper network, we used max-pooling once every 3 convolutional layers to control dimensionality. We implemented drop-out after each fully connected layer. Due to the fact that the network input is non-square (640x128), the stride values are often set to [2,1]. This prevents a situation where one dimension goes down to 1 and the other remains high.

Secondly, we implemented two 1D CNN models - A shallow and a deeper one as well. We paid close attention to the fact that our input is a spectrogram rather than a regular image. Hence, we were not sure that 2D convolutional layers have any positive impact over 1D convolutional layers. It might be the case that 1D convolutional layers that slide over the time axis perform better. Consequently, we built a shallow model with 4-layer 1D CNN + 3 fully-connected layers. Then, we constructed a deeper model with 10-layer 1D CNN + 3 fully-connected layers. Similar to the 2D models, we used batch normalization, max-pooling and drop-out.

Then, we took the RNN hammer out of the toolbox. We decided to implement a naïve straight-forward RNN model (LSTM + fully connected) to attempt classification. As we suspected, the results were pretty horrifying. Our Mel-spectrograms are not really sequential like natural languages.

Lastly, it made perfect sense to combine the two using a CRNN model [9].

Such model introduces a convolutional network where the last convolutional layer is replaced by a RNN. We thought that this model might perform best - local feature extraction of CNNs with temporal summarization of RNNs. We constructed a CRNN model that is built of 3-layer 1D CNN, followed by a LSTM layer and then 2 fully-connected layers. Similar to the 2D models, we used batch normalization and max-pooling. We used drop-out after each layer. After examining the results of the above CRNN, we were surprised to acknowledge the relatively-poor performance, and decided to try another architecture of CRNN. This time, we stacked two LSTMs together. We used a deeper network with 5-layer 1D CNN, followed by stacked LSTMs and then 3 fully-connected layers. We used the same regularization techniques, but instead of using drop-out after every convolutional layer, we used it only after the first one. This decision was made after experimenting with over/under fitting and is further explained in the Discussion section.

2.3 Design

We implemented our project using Google Colab and FMA small data-set. We used Python 3.6 and PyTorch 1.4.0.
Our training time was not too long- the models are pretty straight-forward, and the data size is reasonable.
At first, we tried to forward the validation set as a whole (without breaking into batches), and we ran into difficulties. The main problem was getting runtime error- running out of memory of GPU ("CUDA out of memory"). Using batches of validation set by a data-loader fixed it. Eventually, we used data loaders for both train and validation sets. Our final training process took between 4-25 seconds per epoch.
At the table below we can see the seconds per epoch for each of the models we build. The models have been numbered respectively: (1) CRNN, (2) CRNN Stacked, (3) RNN, (4) CNN 2D Deep, (5) CNN 2D Shallow, (6) CNN 1D Deep, (7) CNN 1D Shallow.

Model	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Sec per Epoch	25	53	3	17	4	9	7

3 Experimental results

We ran each model for 70 epochs using a batch size of 32. For each model, we plotted train and validation accuracy and then plotted train and validation loss. Lastly, we calculated accuracy on the test set. The loss graphs can be found in our Google Colab notebook. In total, we built seven models to attempt verification of our hypotheses: CNN 2D Deep, CNN 2D Shallow, CNN 1D Deep, CNN 1D Shallow, RNN, CRNN, CRNN Stacked.
Listed below are accuracy metrics for our models.

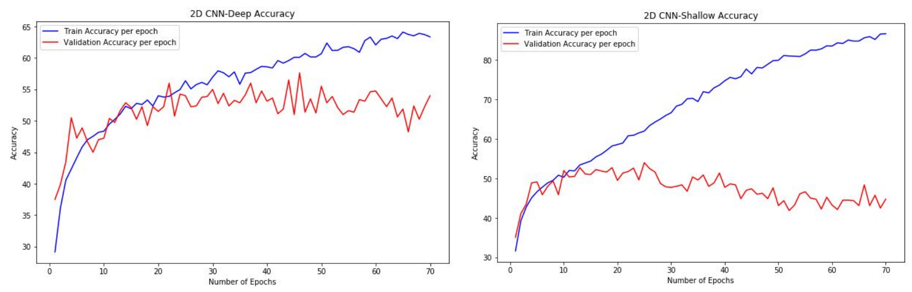

Figure 2: CNN 2D Deep and Shallow accuracy performances

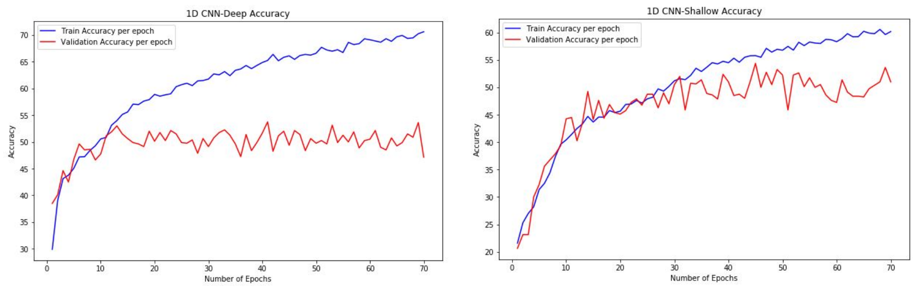

Figure 3: CNN 1D Deep and Shallow accuracy performances

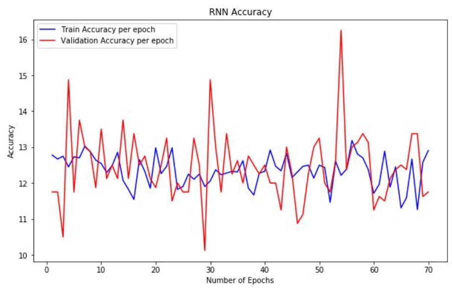

Figure 4: RNN accuracy performances

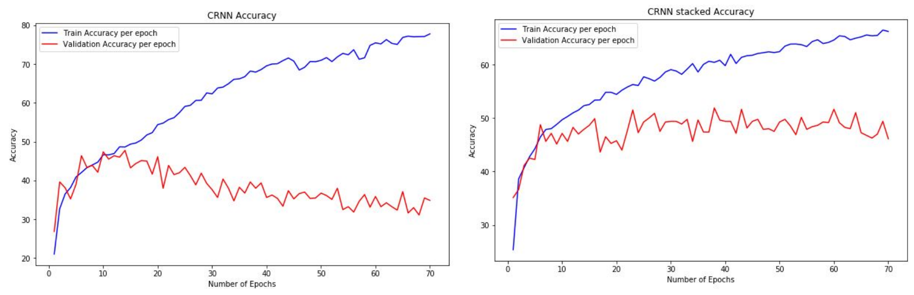

Figure 5: CRNN and CRNN Stacked accuracy performances

Below is a table of all models together, comparing accuracy performances and training time. Like before, the models have been numbered respectively: (1) CRNN, (2) CRNN Stacked, (3) RNN, (4) CNN 2D Deep, (5) CNN 2D Shallow, (6) CNN 1D Deep, (7) CNN 1D Shallow.

Model	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Training Time (sec)	1738	3714	233	1182	321	673	526
Max Validation Accuracy (%)	51.875	51.875	14.75	57.625	54.0	53.75	54.375
Max Text Accuracy (%)	27.125	40.125	11.1	40.375	31.5	33.125	33.125

4 Discussion

After plotting the graphs, some of the results were self explanatory, and some weren't and needed further thinking and re-experimenting.
One of the simplest insights is the over-fit. After dozen epochs we reach over-fit, yet we still ran each model for 70 epochs. These results are easy to explain- as shown in Figure 1, the diversity of the 8 genres mel-spectrograms is not that different. Unlike classification problem such as distinguishing different animals, these unique patterns do not have much difference between each genre. Moreover, the complexity of the shapes in the mel-spectrograms is very low. Our models do not need many epochs to learn the pattern, so we reach over-fit fast.

For the same reason, it is expected that our deep and shallow 2D CNNs show similar validation accuracy, as introduced in Figure 3. Our deep model has a slight advantage in test set accuracy, with 40% compared to 32% of the shallow model. Naïvely, we thought that since deeper networks can detect more complex patterns, it is more prone to over-fit. Surprisingly, the shallow model over-fits faster. At Figure 3 we can see that at epoch 24 we reach over-fit.

When we decided to run 1D CNNs as well, our main line of thought was that convolving over a single axis might produce better results. This is due to the fact that each axis of our mel-spectrograms has a different meaning. In practice, comparing Figure 3 and Figure 4, we can see that the validation accuracy remains pretty much the same. Unfortunately, the shallow and deeper 1D models achieved same lame test accuracy of 33%.

We observe that our 1D models produce less over-fit than the 2D models. Also, they introduce faster run-times.

As can be seen in Figure 5, our RNN model performed poorly. We predicted this bad performance. Although dealing with a time dimension, our data is not really that sequential.

We had great expectations from our CRNN model. We thought that combining CNNs with RNNs will out-perform all other models. We attempted to place different number of convolutional layers before the LSTM. Also, we tried stacking a few LSTMs together with and without drop-out. We delved into thinking whether it is a good idea to place drop-out after convolutional layers to improve our model. There's no clear answer on this matter. In our stacked LSTM, we placed drop-out only after the first layer and not after every layer to avoid under-fit [10]. Our stacked LSTM achieved 40.125% test accuracy, and apparently generalizes better than the first LSTM model that achieved 27.125%. To sum up, although we improved the CRNN, we did not manage to overtake the CNN models. Also, it is important to note that today's top-notch models in music genre classification avoid use of RNNs, such as [4]. This probably emphasizes the idea that mel-spectograms are far from being sequential data.

5 Code

Google Colab link:
`https://colab.research.google.com/drive/12BYSV_ityql8HoBvzfT8x4wVVRQyOyhK`
FMA small dataset before pre-processing:
`https://github.com/mdeff/fma`
FMA small dataset after pre-processing:
`https://drive.google.com/drive/u/0/folders/1-PTQBiz6E53uUa9LebHjds_ZQesRHEqx`

References

[1] Gouyon F., Dixon S., Dance Music Classification: A Tempo-based approach.
[2] M. Haggblade et al., Music Genre Classification.
[3] Mingwen Dong. Convolutional neural network achieves human-level accuracy in music genre classification. CoRR, abs/1802.09697, 2018.
[4] Y. Panagakis, C. Kotropoulos, and G. R. Arce. Music genre classification via sparse representations of auditory temporal modulations. In 2009 17th European Signal Processing Conference, pages 1–5, Aug 2009.
[5] Sahidullah, Md. Saha, Goutam. "Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition". url: https://www.sciencedirect.com/science/article/pii/S0167639311001622?via
[6] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, Xavier Bresson. "FMA: A Dataset For Music Analysis". url: https://arxiv.org/abs/1612.01840
[7] Priya Dwivedi, url: https://github.com/priya-dwivedi/
[8] Brian McFeek, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar,Eric Battenberg, Oriol Nietok. "librosa: Audio and Music Signal Analysis in Python"
[9] Choi K. et al., Convolutional Recurrent Neural Networks for Music Classification
[10] Gal Y., Ghahramani Z., Bayesian Convolutional Neural Networks with Bernoulli approximate Variational Inference. url: https://arxiv.org/pdf/1506.02158v6.pdf