Classifying Music to Spotify Editorial Playlists using Convolutional Neural Networks

Marc Stern
M.S. Data Science
Montclair State University
NJ, USA
sternm2@montclair.edu

***Abstract*—This paper discusses the techniques, methodology, and experimentation for creating a multi-class classification machine learning model to predict and classify songs to Spotify editorial playlists.**

Keywords—Convolutional neural networks, multi-class classification, PyTorch, music playlists, Spotify

I. INTRODUCTION

Music streaming has become the dominant form of music listening over the last decade, largely due to the success of Spotify, the most popular music streaming service in 2022 [1]. One of its focal points as a marketplace is music playlisting, of which there are two types: algorithmic, and editorial. A music playlist consists of a list of songs that play through one after another, giving a listener the experience of a variety of songs by different music artists that are connected through musical or symbolic themes, similar enough to be enjoyed one after another, and yet different enough to keep a user engaged. It is also a popular method for listeners to discover new songs and artists.

While Spotify allows users to create their own playlists, the Spotify official playlists often have tens of thousands to millions of followers, who subscribe to these playlists and are notified on their updates. Thus, for music artists, playlists have become a necessary platform for having their songs discovered by new fans, streamed for royalties, which often lead to ticket and merchandise sales. Playlists have become a mechanism for music artist success. However, getting one's music onto a playlist, without the backing of prior success and popularity, or the finances for advertising new music, can be a challenge. While algorithmic playlists on Spotify are generated automatically for a specific user based on their listening and application engagement history, there are options for artists to submit music to the manually-curated playlists by Spotify's editorial teams.

The goal of this project is to provide artists with a tool that will help them understand the classification and fitness of their songs to editorial playlists on Spotify before pitching them to Spotify.

II. RELATED WORKS

This project is novel, in that there are no exact works to date that predict Spotify editorial playlists for song input. However, there are related works that use the same processes for music recommendation.

Adiyansjah, et al [2] compare convolutional neural network (CNN) modeling with recurrent neural network (RNN) modeling for music recommendation systems. They compare the two types of models, and add genre labels to reinforce music recommendation, which they show helps accuracy. Their models are binary classifiers that compare precision, recall, and area under curve (AUC) scoring methods. They also experiment with having test subjects rate their best model.

Dorfer, et al [3] experiments with feature-combination hybridization systems for music playlist continuation, which refers to music recommendation after a playlist ends. They compare their models to baseline systems such as collaborative filtering matrix factorization, hybrid matrix factorization, neighbors, popularity, and random. They find that their proposed hybrid systems work best, but for different applications.

McInerny, et al describe the music recommender system framework used in Spotify's application, called bandits [4]. The bandits system is heavily based on collaborative filtering, given the vast quantity of data Spotify has at its disposal. It uses typical exploitative methods for suggesting music, while also implementing exploratory and explanatory methods in the user interface of the application allowing users to give Spotify more data. This paper focuses on the context and reward model, which is primarily based on logistic regression. Batches are retrained daily to improve model success over time. Their offline A/B tests show an improvement of 20% in streaming rates when comparing the bandit framework with static explanatory ordering.

Sachdeva, Gupta, and Pudi [5] implement attentive neural networks, which emphasize short term sequences ("next-item prediction"), for music recommendation. They also incorporate song features to enhance their model. To accomplish this, they implement a mini-batch stochastic descent gradient algorithm, and they show that their model outperforms others including RNN, Popular, and more. Their model also finds its best results when using feature tags compared to without feature tags.

Dieleman was another Spotify employee at the time of his experiments, benefitting from the internal Spotify data. He uses CNN deep learning to create playlists based on low and high level musical features and sub genres, as well as similarity based playlists [6].

III. METHODOLOGY

A. DATA COLLECTION

Data collection for this project began with the Chartmetric API. Chartmetric [7] is a music industry data collection platform that contains multitudes of data pertaining to streaming and social media data for music artists. It also includes a history of song selection for music playlists on Spotify. There are thousands of official playlists on Spotify, and only a small subset of them are editorial. Furthermore, of the editorial playlists, only some of them contain a fixed number of songs that are not partly customized for user experience. Filtering out playlists containing keywords such as "top" and "charts," referring to playlists that are based on popularity and not manual curation, and eliminating playlists that are based on cover songs and not original music, 82 playlists remained relevant to this project.

Once these playlists were selected with the correlating Spotify Playlist IDs (URIs), the song IDs (track URIs) could be collected from the Spotify API. This process was replicated several times to collect updated playlist data to increase the overall data count. With the track URIs, the song audio was collected using spotDL's spotify-downloader [8], which searches YouTube for the corresponding song and collects an mp3 file. With the audio stored in a local folder, a proper data set had been created.

The data set at its final stage contained 13,979 songs and 82 playlists. The playlist song counts ranged from 51 to 504, with a median song count of 140 and mean of 170.5.

B. AUDIO TRANSFORMATION

After the data was collected, there were still steps to take to be able to train a convolutional neural network. First, the audio data was not good enough for classification; what was needed was image data.

Using PyTorch's torch audio, the audio was transformed into mel-spectrograms – a feature extraction method for converting audio into images. Specifically, it extracts vectors of logarithmic values for three features: amplitude and frequency over time. These are the foundations of any audio waveform, including music. It does this by using Fourier Transform, a mathematical formula for discerning individual frequencies and their aptitudes at a given moment of time.

Digitized music can come in many formats, and it was necessary to compare identical forms of music. The audio was transformed to: mp3 file type, 44,100 kHz sampling rate, 16 bit depth, and 120 seconds of

time. Red book audio, which is a digital standard used for CD [9], contains the sampling rate and bit depth specifications used in this project. Sampling rate refers to how many samples of audio per second are rendered, and can be thought of as sonic detail, specifically relating to the frequency parameter. Bit depth is related to dynamic range, which we are capturing as amplitude. Mp3 file format is selected to keep storage volume low. Red book audio uses WAV format, so mp3 is considered lower quality, but it is a popular format for audio listening, and more akin to the quality of ogg vorbis, which is used on Spotify streaming. For time-domain consistency, the audio is either truncated at two minutes, or zero-pad-extended if the song is shorter. This is a necessary step for training the model, as the vectors need to have the same shape for matrix multiplication, a fundamental step in neural network functionality.

Once this transformation was complete, the data set now converted class data into numeric values for torch Dataset compatibility, and was mapped to vector data, which was saved and stored locally.

Because there was sufficient data for this project, yet also varying class sizes, it was determined that the data would be split into training, validation, and test sets with a 60/20/20 split. With each new collection of data over time, duplicate data was filtered and new data was randomly assigned at the same proportion into each data set.

With the data in the correct mel-spectrogram format, and split into training, validation, and test sets, the model was able to begin training using torch's mps GPU acceleration. (It is worth noting that torch only had mps Apple M1 chip compatibility as of Spring 2022, about half way into the project, and early iterations were run much more slowly on CPU. This reduced epoch run time from roughly one hour to six minutes. Torch audio however still does not have mps compatibility.)

C. MODEL

The model used was a convolution neural network (CNN), as previously mentioned. This is a model that is good at classification, and specifically with audio data, even after converting to mel-spectrogram vectors. The model was built on top of Velardo's model [10], and it was heavily adjusted for functionality and accuracy improvement.

The model consisted of four convolution layers, each layer consisting 16 channels, kernel size of three, stride equal to one, and padding equal to two. Each layer uses a Rectified Linear Unit (ReLU) activation function and a maxpool before passing the input to the next layer. After the four layers, the data is flattened and a linear function is applied in order to forwardly propagate the data for learning.

Torch's CrossEntropyLoss function receives raw logits, so the final step of the CNN is not an activation function, instead it returns the raw logits of the linear data.

During training, the model used an Adam optimizer to adapt the learning rate for optimizing what had been learned by the model. It also implemented torch's Dataloader object for training and test sets, which shuffle the data every epoch iteration.

During test set prediction, a sigmoid activation function was used to return "fitness" values. The sigmoid function made sense for this use case as a song can exist on one or more playlists with varying "fit," which was the case in the training data.

In order to score the model, accuracy was used as the source of measurement, using the formula:

$$accuracy = \frac{correct\ predictions}{total\ predictions}$$

Accuracy was recorded for top 5, top 3, and top 1 playlists.

IV. EXPERIMENTATION & RESULTS

The final model parameters that were used included: batch size of 16, 12 epochs, and learning rate 1e-5. Each of these parameters were experimented with in hyperparameter tuning one at a time, and these yielded the best results. Other experimentations included length of song, using 60 seconds and 180 seconds, but due to the varying song lengths, 120 seconds was deemed the best. The optimizer did not use weight decay in the final model, but weight decay was experimented with using smaller learning rates and higher epoch counts, but learning was not achieved as efficiently. This part was in response to the fact that after 12 epochs, no more learning was being done, and the model began

to overfit. This was an experiment to mitigate overfitting, but did not prove worthy.

The final training and validation results can be viewed in fig 1. This table records accuracy for top 1 prediction.

Epoch	Training accuracy %	Validation accuracy %
1	8	10
2	35	12
3	58	11
4	68	11
5	73	12
6	76	12
7	77	12
8	78	13
9	78	13
10	78	14
11	78	15
12	78	15

(Fig. 1)

The scores here show that the model continues to train until the end, but tapers off in learning after the first five epochs. Figure 2 shows the results on the test set for top 1, top 3, and top 5 predictions averaged across all classes.

Top 1 %	Top 3 %	Top 5 %
15.49	27.26	47.37

(Fig. 2)

Notably, the test set performs better than the validation set. This is a good sign that the model is not overfitting, and is working as well as it can, and that the training and validation sets are reliable. The overall score would not be promising in the context of binary classification, as it would not be better than random classification, but in the case of having 82 classes, it scores significantly better than a random classifier, which would score at a rate of 1/82 (1.22%) for top 1.

The top performing classes were *Gentle Rains*, which scored at 100% across all three categories of accuracy, and *Nightstorms*, which scored at 100% for top 5 and top 3, while getting 72.5% correct for top 1. 11 classes in total scored higher than 80% accuracy for top 5, while 13 classes scored 0% for top 5. This shows that the model excelled in some areas, and failed in others, with the full range of accuracy on display.

V. DISCUSSION

The results show that this model still needs work, but has certain strong qualities to it. As one might expect, the classes that performed the best were also those that had the most training data. It is encouraging to see that the model not only works for those classes, but it works extremely well, often as high as 100%. The most notable success of the results are the two classes that are not actually music, but instead ambient soundtracks, including rain and thunderstorms. From this, it could be concluded that the model understands very accurately what is music and what is not, and it could be repurposed as a binary classifier to identify music versus non music. The nature of having 82 classes makes prediction much more challenging of a task. Overall, all playlists relating to music that could be considered relaxing, such as soft or background piano music, the aforementioned ambient songs, or those curated for spas and specifically relaxation perform the best. This makes sense as those songs often are missing percussive elements such as drums, giving the model an easier opportunity to identify related classes.

The model struggles with other forms of music that are more popular styles, such as pop, hip-hop, country, rock, and more obscurely titled playlists. While this project is not specifically a genre classifier, it is worth noting that genre plays a big part of music playlisting. Oftentimes songs of these genres can have similar musical elements, which appear similar in the context of a mel-spectrogram. These playlists also contained less training data, resulting in lower performance.

Based on the above comments, a reasonable hypothesis for a future project would be that this model would work well identifying fewer classes like: “Is a song relaxing?”, or as previously mentioned, “Is a song music or ambient sound?” That was not the specific goal of this project, but worth pointing out what the model excelled at, and how to exploit that in future work.

VI. CONCLUSION

Due to the varying results of this model, there are many ways in which further iterations of the model could be improved.

The main source of result fluctuation was due to lack of data. For optimal convolutional neural network models, the distribution of data per class should be even, and in the case of this project it was not. Additionally, the more data the better, and collecting data in the thousands, compared to the hundreds of class data collected in this project would be ideal. Based on the results of this model, it could be expected to see vast improvement with more data.

Data augmentation could be a helpful method for artificially creating more data from the data collected in the current data set. Due to time constraints and the difficulty of switching back and forth between PyTorch versions (after torch mps compatibility became viable, but not yet for torch audio) made this a future step for improving model accuracy.

Furthermore, other methods of feature extraction could be experimented with in addition to mel-spectrograms. These require more data points for classes, and run into similar aforementioned issues with torch audio compatibility.

In general, future iterations of the model could have a stronger emphasis on hyperparameter tuning and experimentation. This project focused on data collection, but never reached a point of having enough data. With enough data, these other methods would be appropriate to further the improvement of the model accuracy.

One step for evaluating the model that would make sense is to experiment with people, having them rate the model output. After all, the playlists in this project are specifically filtered out to be manually curated, and music is a subjective artform. This project attempts to objectify what is otherwise considered subjective, but it would be interesting to see how people score the model’s accuracy compared to a binary (correct or incorrect) system, especially when considering songs can be on multiple playlists.

VII. ACKNOWLEDGEMENTS

I would like to thank my advisor throughout this project, Dr. Boxiang Dong, for his guidance through the project providing feedback at all stages, especially in the early stages of conceptualizing the project and providing related works to study to help aid in methodology.

I would also like to thank Dr. Nitin Madnani for his technical help in critical stages of the project including using SpotDL and running PyTorch with mps GPU acceleration, as well as his general advice in improving classifier accuracy.

VIII. REFERENCES

1. D. Curry, “Music Streaming App Revenue and Usage Statistics (2022),” Business of Apps, May 4, 2022. [Online]. Available: https://www.businessofapps.com/data/music-streaming-market/
2. Adiyansjah, Alexander A S Gunawan, Derwin Suhartono, “Music Recommender System Based on Genre using Convolutional Recurrent Neural Networks,” Procedia Computer Science, Volume 157, 2019, Pages 99-109. [Online]. Available: https://doi.org/10.1016/j.procs.2019.08.146
3. Vall, A., Dorfer, M., Eghbal-zadeh, H. *et al.* Feature-combination hybrid recommender systems for automated music playlist continuation. *User Model User-Adap Inter* **29,** 527–572 (2019). https://doi.org/10.1007/s11257-018-9215-8
4. James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Grunso, Rishabh Mehrotra, “Explore, exploit, and explain: personalizing explainable recommendations with bandits,” Association for Computing Machinery. [Online]. Available: https://doi.org/10.1145/3240323.3240354
5. Noveen Sachdeva, Kartik Gupta, Vikram Pudi, “Attentive neural architecture incorporating song features for music recommendation,”

Association for Computing Machinery. [Online]. Available: https://doi.org/10.1145/3240323.3240397
6. S.Dieleman, “Recommending music on Spotify with deep learning,” Aug 5, 2014. [Online]. Available: https://benanne.github.io/2014/08/05/spotify-cnns.html
7. Chartmetric API. [Online]. Available: https://chartmetric.com/
8. SpotDL, “spotify-downloader.” [Online]. Available: https://github.com/spotDL/spotify-downloader
9. Red Book (audio CD standard).” [Online]. Available: https://en-academic.com/dic.nsf/enwiki/26666
10. V. Velardo, “PyTorch for Audio + Music Processing, The Sound of AI. [Online]. Available: https://www.youtube.com/playlist?list=PL-wATfeyAMNoirN4idjev6aRu8ISZYVWm