# Content based music recommendation with deep learning

Wanhe Zhao
Pittsburgh, PA
wanhez@andrew.cmu.edu

Junyan Pu
Pittsburgh, PA
junyanp@andrew.cmu.edu

## 1 Introduction

When music streaming service providers are giving music recommendation to their users, a wide varieties of machine learning and deep learning algorithms have been used. Today's music recommenders such as Spotify, Pandora and iTunes have made a great progress in learning users' taste based on streaming histories and meta-information about the songs to minimize user effort to provide the recommender the feedback. This project is inspired by lifehouse method project, an innovative portraiture system, launched by one of members from the Who, Pete Townshend. He believed that "music is our only hope. I believe music can reflect who we really are - like a mirror"[1]. Therefore, we believe that building a personalized song recommendation system is such an exciting field that could help people to rediscover their own taste.

In this project we investigated in some of the dominant recommendation algorithms to train our own deep learning model to learn either the features of songs or to gain insights from user datasets. Specifically, we implemented two dominant recommendation algorithms: collaborative filtering as our baseline model and content-based CNN model and did an empirical comparison.

For collaborative filtering model, we collected 7,377,304 entries from The kkbox dataset. The dataset contains extracted features from user's usage history, including songs they listened to, features of songs (name, style, artist, length, etc.), users' information (name, age, etc.), and a target (response variable) 0/1 indicating whether the user listen to the song again within a month after user first listened to the song. Using collaborative filtering model, we are able to directly make predictions on whether to recommend a song to a user by looking at the 0/1 output of the model.

For content-based CNN model, we collected total songs of 141,453 from Million songs database (MSD)[2], and each song is represented by a spectrogram based on its audio features, including chromas and timbre. For output label, we use LastFM dataset[3] that has similarity score for each pair of songs, which links to MSD by track ID. After this CNN model, we are able to gain the similarity score between song pair, so we further build our recommendation algorithm based on the similarity score, and our final goal is to output a content-based similarity playlist at user level.

## 2 Background

We first used collaborative filtering method to approach music recommendation problem. This is our baseline model, which provides an empirical result for comparison with our content-based model. We implemented our baseline algorithm on the kkbox dataset for music recommendation [4] using trainable embedding and neural network [5]. The predictors we use are extracted user/song information, including user/song pairs, features of songs, and users' id, with a target (response variable) 0/1 indicating whether the user listen to the song again within a month after user first listened to the song. In this model, we send member id and song id into trainable embedding and project them into a 64 dimension vector space. Then, we take the dot product of embedded member id and embedded song id as a new feature (user, song pairs). After these two steps, we feed in

Submitted as Topics in Deep Learning (10-707, Spring 2019) Project Final Report. Do not distribute.

all features into a neural network with 128 hidden units with reLU activation function. Then, we apply dropout with rate 0.5 and use linear mapping with softmax to map to the target (response variable). For our baseline model using predictors that includes only extracted user/song information, we achieved 76.79% accuracy on the training set, 72.01% accuracy on the validation set, and 79.48% accuracy on the test set. The performance shows that it is effective to perform music recommendation using only user information and extracted song features. The advantage of collaborative filtering method is that the output of our result directly leads to the final decision of recommending the song or not to recommend, with no further algorithm required. However, the recommendation result of our baseline approach highly depends on user's usage and user's previous exposure to songs, instead of the music content.

# 3 Related Work

## 3.1 Collaborative filtering

This a traditional approach to power recommendations. The historical usage data is used for determining the users' preferences. This method can be viewed as a sequence prediction problem, and the intuition is that if two users with similar history streaming data, the system will recommend the content based on these information. However, the main drawback for pure collaborative filtering approach is content-agnostic and the reliance on usage data. This leads to the cold start problem that new content is not being recommended, and also the recommender is bias towards popular songs as there is more usage data available for them [6].

## 3.2 Content-based Deep learning Network

Music could also be recommended based on available metadata: lyrics, text mined from music reviews, the artist, album, year of release, and audio signal. The audio signal deep convolutional neural networks is successfully used by The Echo Nest, a music intelligence platform company acquisited by Spotify. Despite quite a large semantic gap between music audio, it is a promising idea to learn users preference through audio signal. Content-based recommendation has been tackled by many groups previously. One of the approaches trains a regression model to predict the latent representations of songs that were obtained from a collaborative filtering model [7]. This method is able to provide recommendations even if no user data available, and generate playlists through the filters in the network that are picking up different properties of the audio signals. McFee et al. has defined an artist-level content-based similarity measurement using bag-of-words [8].
In this project, inspired by the song pair CNN architecture used by Balakrishnan in [9], our model builds a recommendation system based on similarity score. Our CNN model uses the CNN parameters from [10] as a starting point.
As for data preparation, YMG Costa [10] suggests to down-scale the high-resolution spectrogram images since CNN model does not deal quite well with high resolution image.

# 4 Methods:

To compare with collaborative filtering algorithm, we implements content-based CNN model, which requires no user information or existing user streaming history, but instead focus on building recommendation system based on song patterns, audio structures and similarity scores. There are some previous works on content-based recommendation, but song similarity score prediction hasn't been used in content-based recommendation system so far as we seen. We draw an inductive bias that part of the songs people like have similar audio features or, in other words, look similar on spectrograms. Therefore, we initiate a similarity score based recommendation system in our project here.

## 4.1 Dataset and Feature Engineer

After merging between MSD and LastFM, we collected 127,653 song pairs with imbalanced similarity song pairs, and 13,750 balanced set for testing. 110,085 song pairs with imbalanced similarity score and 11,594 balanced for validation set. Note that training and test set are split based on LastFM train and test set, which ensures there are no same songs trained before seeing them in validation sets.

The spectrogram is generated from the first 60 seconds of each song, and is down-scaled to 40*120 pixels as shown in Fig.1. Our CNN model takes in two numpy arrays transformed from spectrograms for each song pair as input. We then discretize similarity score score to 1, 0 with threshold 0.5 as our y label in our training, with 1 indicating that two songs are similar. Note that we original tried to use 64*120 pixels spectrogram, but a large portion of it contains blue background. Therefore, we decide to concentrate on the lower part of the spectrogram that represents most of the song features.

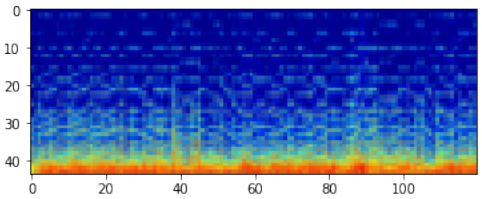


Figure 1: Down-scale spectrogram (40*120 pixels)

### 4.2 Convolutional Neural Network

The deep neural network architecture used in this project has three fully connected convolution layer with sigmoid activation function and followed by 2*2 max-pooling layer. After convolution, we have two flattened layers output vectors ($h^1_{final}$ , $h^2_{final}$). Then, we concatenate two vectors and their dot product together as shown successful in the experiments with three concatenation strategies of two songs flatten layers by Balakrishnan in [9]. Then we use two layers of feed-forward neural network with 100 and 30 hidden units with corresponding sigmoid and relu activation functions. We use binary cross-entropy as our loss function. The CNN model will output the similar score of two songs, ranged from 0 to 1. Detailed is shown in the Fig 2 and 3.

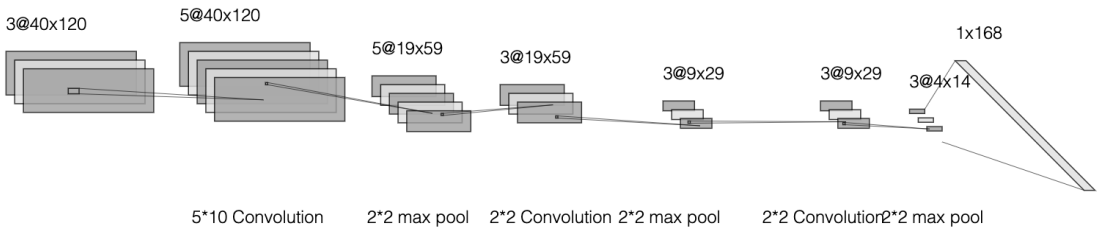


Figure 2: The deep Convolutional Neural Network architecture

We tried to use LSTM to take into account of the fact that audio data is sequential in nature, but the model with LSTM performs quiet unstable. What's more, we tried to make stride size wide in time and narrow in frequency, but this makes the output convolutional layer narrow in width and wide in height. If we want this model to learn the sequential changes, we may need to make the progress of learning in horizontal level slower. Thus, we further tried out stride size bigger in height, but it didn't give us an optimal output. Finally, inspired by YMG Costa [10], we tried both squared kernels and the kernels wide in time and narrow in frequency. Our final model with kernel size (5,10) on the first convolutional layer and square kenel with size 2 on the last two convolutional layers work well in our experiments.

#### 4.2.1 Train on Imbalanced data

We fit the content-based CNN model on an imbalanced dataset, with the proportion of label 0 and 1 equals 9:1, and evaluate the performance on a balanced testing set. Specifically, we split the training data randomly into training set and validation set, and feed them into the model. In order to improve the sensitivity of the model, we tried two approaches to deal with the potential problem of imbalanced

data: adding sample weights and adding class weights. Through these two approaches, we expect to have higher sensitivity in our prediction result. Then, we use a balanced unbiased test set to evaluate the performance of the model.

For sample weights, we use Kernel density function with bandwidth = 0.2 to approximate the probability density function of the similarity score (response) in the training set. Thus, each instance is weighted by the $\frac{1}{pdf}$ value.

For class weights, we assign the class weight according to the label percentage. We assign weight 1 to label 0, and assign weight 10 to label 1. This is equivalent to say that the model treats instances with 0 as 1 instance while it treats instances with 1 as 10 instances.

#### 4.2.2 Train on Balanced Data

Additional weights are not added for balanced data. For balanced data, we fit the model with balanced train set, validate on validation set, and evaluate the performance of the model using an unbiased test set.

### 4.3 Recommendation algorithms

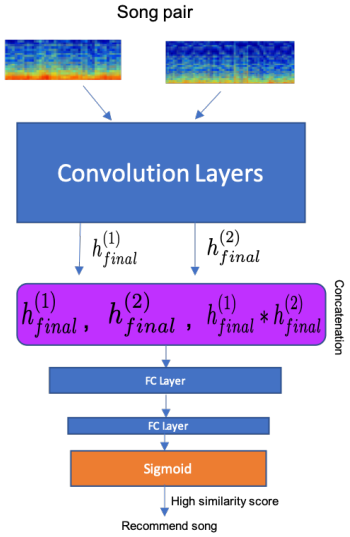


Figure 3: An overview of the recommendation system scheme

After training our song similarity model, we plan to construct a recommendation system to generate a playlist that will reflect users taste. We collected some Spotify users past three months streaming histories. Based on their top five most streamed songs, we pair it with new song from our Million Song Dataset. If our model predicts high similarity score between our song and their most streamed song, then we will recommend to our user.

## 5 Results

### 5.1 Imbalanced data

Using imbalanced training, imbalanced validation data, and balanced test data, our model achieve 92.87% training accuracy, 84.14% validation accuracy, and 85.15% test accuracy. The Fig4 shows the accuracy curve versus training epoch, and the normalized confusion matrix for our results on validation set, and test set.

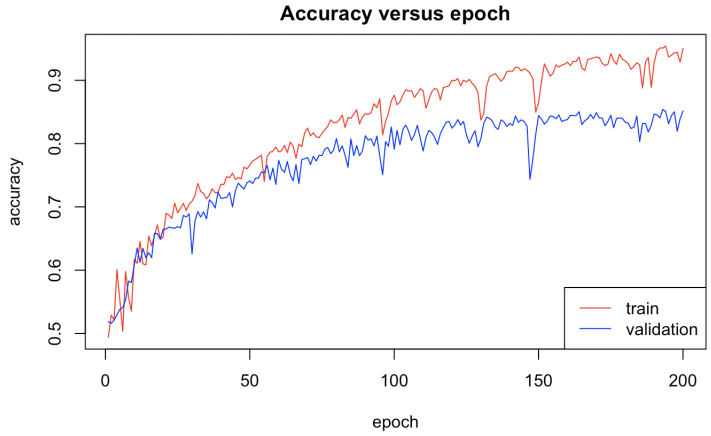


Figure 4: Accuracy on Imbalanced Data

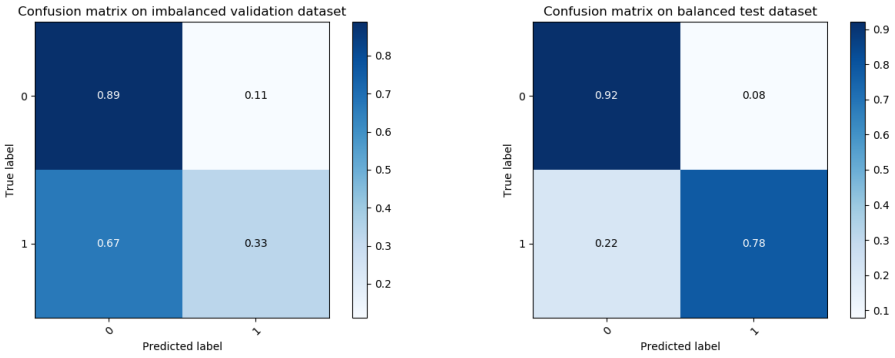


Figure 5: Confusion Matrix of imbalanced validation set(left) and balanced test set(right)

### 5.2 Balanced data

During training, the training classification accuracy continues to improve during 200 epochs, but the validation set will only improves to a point around 62% and converges, before our model starts to overfit. As shown in Fig.6.

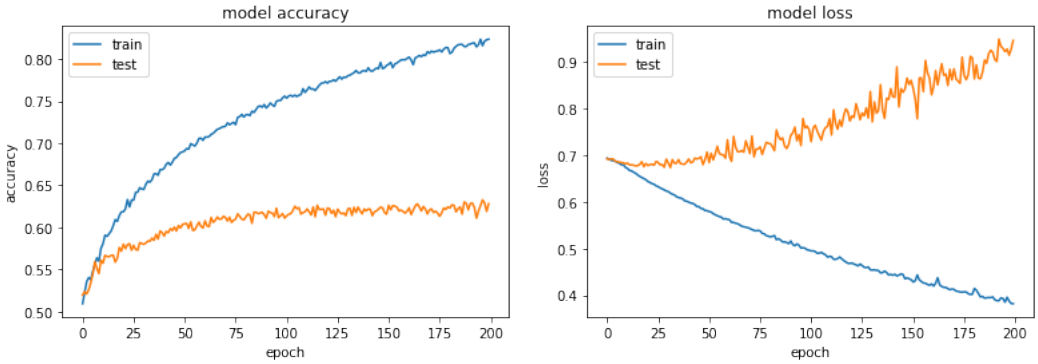


Figure 6: Accuracy and loss plots of balanced train and validation sets

# 6 Discussion and Analysis

## 6.1 Constraints

### 6.1.1 Assumptions

This recommendation system is built based on the assumption that people tend to like songs with similar audio features rather than similar lyrics or similar artist style. Another assumption we make is that the first 60 seconds of a song have enough representation of a song audio features.

### 6.1.2 Interpretability and computational cost

The interpretability of content-based model is not as good as collaborative filtering, and also has high computational cost. We need to first generate image representation of an audio, run it in CNN model, and then using recommendation algorithm to create a recommended playlist. In comparison, collaborative filter is more direct and easy to carry out, since the output is directly link to whether to recommend a particular song.

### 6.1.3 Song representation

Our spectrograms are not precise representation of a raw audio. we only have access to limited audio features that extracted by MSD but not the real raw audio samples. This will introduce noises into our data.

We down-scale spectrograms by 20%, but YMG Costa [10] only down-scale by 50%, which gives a promising result. This low resolution of spectrograms may introduce extra noises. Since data preparation takes a long cycle, we will try regenerating song spectrograms in higher resolution in our future work.

Even though several papers mentioned the advantages of using spectrograms to represent a song, there is still some limitations. As said in Rothmann D: "A particular observed frequency in a spectrogram cannot be assumed to belong to a single sound as the magnitude of that frequency could have been produced by any number of accumulated sounds or even by the complex interactions between sound waves such as phase cancellation. This makes it difficult to separate simultaneous sounds in spectrogram representations."[11] For example a spectrogram may not give an accurate representation of a mono version beatle song.

## 6.2 Performance

### 6.2.1 Imbalanced Set

In the imbalanced set, the class ratio for label 0:1 is 9:1. If guessing label 0, we could achieve accuracy of 90% with sensitivity equals 0. By adding class weights, we effectively increase the sensitivity of the model to a reasonable level. Specifically, even the training set and validation set are imbalanced, our content-based model still effectively captures traits in order to make predictions with both high sensitivity and specificity. According to our performance on the balanced test set, with ratio 1:1 for label 0 and 1, our test accuracy 85.15% is significantly higher than the base ratio 50%.

### 6.2.2 Balanced Set

To train CNN model on balanced sample, we collect more more data. The performance of the content-based CNN model could have improved more if we could have avoided overfitting during training process as shown in 6. Our validation accuracy converges at 62% and training accuracy is 87%.

### 6.2.3 Performance Discussion

The potential problem of noisy data makes it harder for the content-based model to generalize well on unseen dataset. One possible explanation for the underperformance may be that the spectrogram are noisy, such that our model tends to capture the noise in the training data more than the actual underlying structure. Another possible explanation is that the our model is still unstable. The

performance of the our CNN model largely depends on the quality of initialization. Therefore, increasing the resolution, denoising the dataset, and improving initialization methods are possible improvements we can make in the future.

### 6.3 Recommendation algorithms

To complete a recommendation system, we not only need a better model performance, but also need to increase our dataset song coverage. Since we only have access to the audio features of 141,453 songs. We may even don't have access to the audio features of users' most streamed song to generate a spectrogram for running a cnn model prediction.

## References


[1] Pratt, C., "Lifehouse Method" *Pete Townshend's website.* Available at : https://petetownshend.net/musicals/lifehouse-method.

[2] Million Song Dataset, official website by Thierry Bertin-Mahieux, Available at: https://labrosa.ee.columbia.edu/millionsong/.

[3] LastFM Dataset, Available at: https://www.last.fm/.

[4] Kaggle contributors, "WSDM - KKBox's Music Recommendation Challenge." *Kaggle* (2018). Available at: https://www.kaggle.com/c/kkbox-music-recommendation-challenge.

[5] Kaggle contributors, "Beat KKBox Benchmark without using metadata" *Kaggle* (2018). Available at: https://www.kaggle.com/lystdo/beat-kkbox-benchmark-without-using-metadata-0-62.

[6] Dieleman, S., "Recommending music on Spotify with deep learning" *Sander Dieleman's blog.* (2014). Available at: http://benanne.github.io/2014/08/05/spotify-cnns.html.

[7] Oord, A.v.d, Dieleman, S. , Schrauwen, B., "Deep content-based music recommendation" *NIPS Proceedings.* (2013). Available at: https://papers.nips.cc/paper/5004-deep-content-based-music-recommendation.pdf.

[8] Brian M., Luke B., and Gert R. G. Lanckriet., "Learning content similarity for music recommendation." *arXiv.* (2011). Available at: https://arxiv.org/abs/1708.02182.

[9] Balakrishnan A., Dixit K., "DeepPlaylist: Using Recurrent Neural Networks to Predict Song Similarity." (2016). Available at: https://cs224d.stanford.edu/reports/BalakrishnanDixit.pdf.

[10] Yandre M.G. Costa, Luiz S. Oliveira, Carlos N. Silla Jr, "An evaluation of Convolutional Neural Networks for music classification using spectrograms", (2016). Available at: https://doi.org/10.1016/j.asoc.2016.12.024.

[11] Daniel Rothmann, "What's wrong with CNNs and spectrograms for audio processing?" (2018). Available at https://towardsdatascience.com/whats-wrong-with-spectrograms-and-cnns-for-audio-processing-311377d7ccd.