Deep Convolutional Autoencoder for Genre Prediction and Content-Based Music Recommendation

Elliot Tower
etower@cs.umass.edu

Ivan Vegner
ivegner@umass.edu

Abstract

The problem of music recommendation is a challenging one for machines to solve, due to the subjectivity of music. Traditional music recommendation relies upon collaborative filtering, which bases recommendations on the listening patterns of other users with similar tastes. We undertake the more challenging problem of content-based recommendation (CBR), which recommends music based on sonic similarity. We present an autoencoder-inspired architecture for CBR, which is trained on genre classification. The architecture compresses the song audio into a low-dimensional latent space, which is used for genre prediction. We achieve competitive results on genre classification, and the resulting architecture can be used to compress arbitrary songs into the learned latent space, with recommendations consisting simply of finding nearest latent space neighbors.

1. Introduction

In this paper, we tackle the problem of content-based recommendation (CBR) for music. Traditional music recommendation depends largely upon collaborative filtering [11], a method which predicts a user's future music listening patterns based off previous listening patterns, along with those of other users with similar tastes. This method is effective for recommending existing music which is popular, but fails to recommend new or smaller releases which do not have previous listening data off which to base recommendations.

CBR mitigates this problem by analyzing the raw audio data and extracting higher level audio features, which are then used for recommendations. This results in recommendations which are sonically similar, rather than songs which have similar listening histories–but may sound nothing alike. CBR is an emerging field of research, and companies like Spotify extract considerable value from intelligent song recommendation. In this paper, we perform CBR not just to recommend songs, but also create playlists based on similar song features.

1.1. Task Overview

Our problem can be stated as follows: given the raw audio data from a song, extract higher level features, and use these features to recommend similar songs to play next for the user. The recommendations from a given song will exhibit sonic similarities to the that song, more so than randomly selecting a song from the same genre.

We use the Free Music Archive (FMA) dataset, which contains 30 second audio clips from over 100,000 songs, as well as metadata such as artist name and genre. Using the FMA dataset as input, our network performs two tasks:

1. Genre classification using the genre-balanced FMA-small dataset. At 8,000 songs (8 genres), this dataset is 8x larger than GTZAN, which is the second most popular dataset used for genre classification.

2. Playlist generation and song recommendation using near-neighbors in compressed feature space

The 8 genres in the FMA-small dataset are as follows:

Electronic	Pop
Experimental	Rock
Folk	Instrumental
Hip-hop	International

2. Background

Music recommendation and playlist generation are both rapidly expanding fields, but have shown to be difficult to evaluate, due to the subjective nature of music. There is much research being done in both areas, but currently there is no agreed upon standard of evaluation for either [17]. Quantitative methods of evaluation rely on assumptions which do not commonly hold in practice, and often reduce it to an information retrieval task [15].

One such method of evaluation is taking a dataset of existing playlists and trying to predict the next song in the playlist, evaluating whether the prediction was correct. However, these such datasets contain large numbers of playlists scraped from the internet, and they may not even

be intentionally ordered, which renders the evaluation extremely noisy.

Another method of evaluating generated playlists is the Semantic Cohesion test, which evaluates playlists based on the frequency of metadata co-occurences: songs by the same artist, of the same genre, etc. However, research shows that the assumption of coherence characterizing playlist quality may not be justified, as users often prefer playlists which are more diverse [1].

Because it is so difficult to measure the quality of recommendations, some works cite genre classification accuracy as a comparison benchmark [5] [7].

3. Prior Work

3.1. Music Recommendation

Schedl (2019) provides an overview on recommender systems and the history of deep learning applications to music recommendation [15]. Dieleman (2013) was one of the first uses of deep learning for music recommendation, using a convolutional neural network (CNN) to analyze raw audio in the form of spectrograms [5]. This approach has become a standard and is used in many subsequent works, with increasingly deep architectures and some additional pre-processing steps such as data augmentation [2]. Dieleman and other early works trained on the Million Song Dataset (MSD), but recent works have shifted to Free Music Archive (FMA) due to the availability of copyright-free raw music audio.

Music recommendation differs from traditional recommendation (e.g., for movies or books) in a number of important ways. Firstly, the duration of a song is much shorter than that of a movie or book, and a user can tell much quicker if they do not like a recommendation. Second, there is much more sparsity: the amount of music on a given streaming service is in the order of tens of millions of songs, compared to a few thousand movies on Netflix, for example. Collaborative filtering simply does not work as well with music due to the sheer number of songs.

Music is also unique in that the listening order matters significantly, and it can evoke strong emotional reactions. Emotion aware music recommendation systems such as Deng (2015) use deep learning to match a user's mood to the mood evoked by other songs for past listeners [4]. The predominant method of song recommendation is by finding near-neighbors in some latent space, but some have found success with other methods such as deep belief networks (DBN's) and weighted matrix factorization (WMF) [15] [5].

3.2. Genre Classification

Jiminéz [10] provides an overview on genre classification, comparing deep learning architectures such as CNN, RNN and hybrid approaches, tested using the GTZAN dataset (10 genres, 100 songs per genre). Jiminéz found that convolutional neural networks were favorable for this task on due to their ability to efficiently extract features from the 2-d spectrogram inputs [10].

This formulation of audio as a 2-dimensional spectrogram allows CNN's success in visual recognition tasks to apply for audio learning tasks as well [12]. Kim (2020) also finds CNN's to be effective for this task, although more research is needed to determine generality in the context of deep transfer learning (e.g., applying to new tasks or datasets) [11]. This work also notes that 1-d variants of CNN architectures were suggested for learning features from raw audio for end-to-end learning, but not many were successful in music classification tasks [11].

Different datasets for music genre classification find greatly different results. On the GTZAN dataset, Li was able to achieve 84% in 2010 [14], and Choi was able to achieve 89.8% in 2017 [3]. However, the FMA-small dataset is considerably larger and due to more variety in song signatures, state of the art performances are less accurate. As FMA-small was created in 2017, related results are somewhat limited. Lee was able to achieve 51.2% accuracy using transfer learning CNN in 2019 [12]. Also in 2019, Bian et all achieved a new state-of-the-art performance of 68.9% accuracy using DenseNet and data augmentation, and notably used 1-d convolution [2]. Data augmentation included pitch shifting, and overlapping time windows.

Model	Author	Accuracy
Transfer learning CNN	Lee, 2019	51.2
CNN w/ data aug.	Bian, 2019	64.7
ResNet w/ data aug.	Bian, 2019	66.7
DenseNet w/ data aug.	Bian, 2019	68.9

Table 1: Comparison of previous state-of-the-art models on FMA-small dataset. Note that Bian (2019) uses data augmentation, which generates additional data to train on.

4. Approach

4.1. Data preparation

The FMA-small dataset had a number of issues, from missing metadata to different sample rates for the constituent songs. Before training, we had to extensively clean the data and convert each of the files to a uniform sampling rate for use with the Librosa[1] plugin for Python. Each song is clipped to 30 seconds in length, and then the raw audio is converted to a mel-spectrogram. Mel-spectrograms are a compact representation of the frequency makeup of the song, computed by applying windowed short-term Fourier transforms to the waveform. The mel aspect of the spectro-

[1] https://librosa.org/doc/latest/index.html

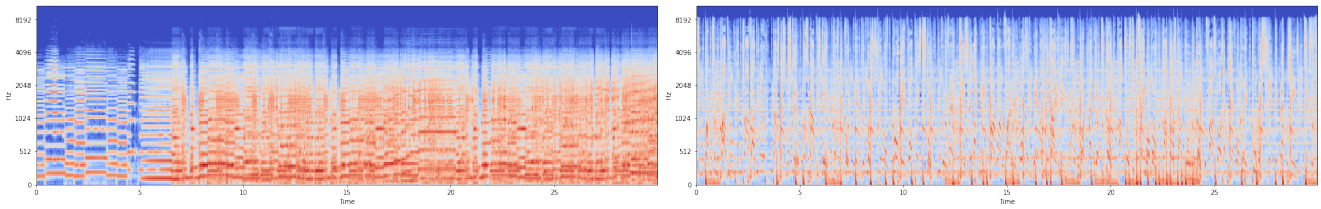

Figure 1: Mel-spectrogram of a jazz song (left) versus a hip-hop song (right)

gram refers to the quasi-logarithmic scale of the frequency dimension. This is a standard process in music information retrieval (MIR) research [17], as converting to a mel-scale accounts for the fact that humans do not perceive sound frequencies in a linear fashion (e.g., differences between low-frequency bass notes are more noticeable than similar differences between high-frequency notes).

Music from different genres exhibited clear differences in patterns of intensity over time, as seen in Figure 1. The song audio is transformed into a 2D image, with time on the x-axis and frequency in Hertz on the y-axis. The amplitude of a given frequency y at a particular point in time x is given by the pixel color at point (x, y) – the redder, the higher the amplitude. This representation has been found amenable to conventional CNN approaches [10].

4.2. Architecture

Given the difficulty of evaluating music recommendations quantitatively, and lack of a standard to do so, we choose to evaluate playlists and song recommendations qualitatively. However, while the playlist generation and song recommendation is the ultimate goal of the architecture, it is not the only application thereof.

Our architecture is inspired by an autoencoder paradigm. It consists of a deep CNN "encoder" compressing the input spectrogram into a low-dimensional latent space, followed by a relatively shallow fully-connected sub-network that predicts the genre of the song ("predictor").

Our approach for the predictor network is heavily influenced by Dieleman and Schrauwen [5, 6]. While the overarching goal of our endeavor is music recommendation, the purpose of the genre predictor head is to provide a quantitative proxy metric for the usefulness of the latent space for song recommendation. Genre classification is a more well-studied problem than song recommendation, and it can be expected that any good latent space of song features will be heavily clustered by genre – after all, the most basic form of song recommendation is recommendation by genre. Thus, using the latent space of the model to predict the genre of a song and evaluating the genre classification accuracy is a suitable proxy metric for the quality of the latent space.

After jointly training the architecture, we use the encoder part of the model to:

1. Choose an arbitrary 'seed' song for recommendations, and compress it into the latent space.

2. Find latent space near-neighbors among songs to find songs that are most similar to this 'seed' song.

3. Create playlists by grouping together songs which are close in a few latent space dimensions. These latent space dimensions form a playlist "signature", which we can compare to other signatures and can be used for extending playlists further by adding songs that most match the playlist signature. Similar signatures can be computed for albums by their constituent tracks, and even users by their libraries.

4.3. Encoder

The input to the model is of shape $(N, 1, M, T)$ – that is, N different song spectrograms, each separated into M mel-frequencies and comprised of T timesteps. The singleton dimension can be inserted to make the shape of the spectrogram analogous to that of an image with 1 channel. However, it is important to note that unlike in an image, the M and T dimensions do not carry the same meaning.

The choice between 2-dimensional and 1-dimensional convolutions for mel-spectrogram inputs is a point of contention in literature. All approaches convolve along the time axis, which allows for a higher receptive field that can account for the sonic qualities of the whole song. However, some models also convolve along the frequency dimension. Ulyanov (2016) notes that this approach can lead to invariance to pitch shifting, which is a favorable quality.

While 2-d convolutions have more parameters and are able to capture patterns in both dimensions, some argue that unlike in images, the dimensions of spectrograms do not carry the same meaning and thus 2-d convolutions are inappropriate [2] [16]. We initially started with 2-dimensional convolution because in theory, 1-dimensional convolutional filters are a subset of 2-d convolutional filters, and thus theoretically should be able to be learned if it is expedient for the network to do so. However, given the stellar performance of Bian (2019) with 1-dimensional convolution [2], we evaluated 1-dimensional convolutions as well.

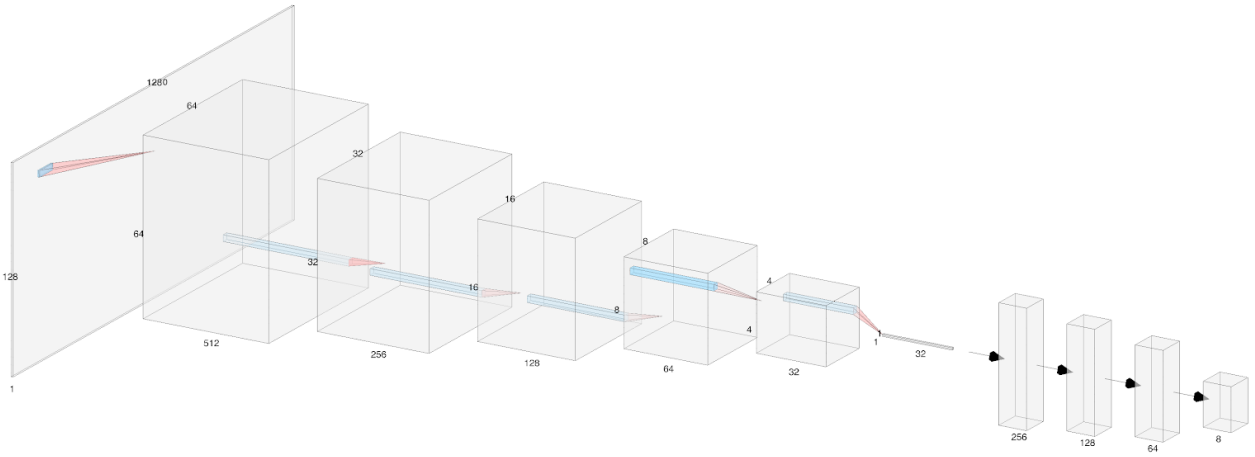

Figure 2: Architecture of the network.

Layer	Channels	Filter dim	Stride	Pad
Conv+ReLU	512	(3,11)	(2,5)	(1,5)
Conv+ReLU	256	(3,3)	(1,1)	(1,1)
MaxPool	-	(2,2)	(1,1)	-
Conv+ReLU	256	(3,3)	(1,1)	(1,1)
MaxPool	-	(2,2)	(1,1)	-
Conv+ReLU	128	(3,3)	(1,1)	(1,1)
MaxPool	-	(2,2)	(1,1)	-
Conv+ReLU	64	(3,3)	(1,1)	(1,1)
MaxPool	-	(2,2)	(1,1)	) -
Conv+ReLU	32	(3,3)	(1,1)	(1,1)
MaxPool	-	(2,2)	(1,1)	-
Conv+ReLU	32	(2,4)	(1,1)	(0,0)
MaxPool	-	(1,5)	(1,1)	-

Table 2: Structure of the encoder

The encoder is comprised of 7 stacked convolutional and max-pooling layers, with the ReLU nonlinearity applied to the convolutional outputs. The exact structure is given in Table 2. The first convolutional layer uses large filters to rapidly compress the spectrogram's time dimension, extracting low-level patterns from wide swaths of time at once. This serves to rapidly reduce the dimensions of the input, and is effective because the information in a spectrogram is highly redundant from one point in time to the next. Each subsequent convolutional layer decreases the number of filters (from 512 initially to $F = 32$ latent space dimensions) while reducing the size of the input spectrogram in both the frequency and time dimensions. Thus, each consecutive module extracts high-level features from the input while becoming increasingly more agnostic to the specific frequencies and time position of the features. The Max Pooling operations contribute to this effect, by further reducing redundancy across adjacent frequencies and time windows. The weights of all convolutional layers are initialized using Kaiming initialization [9], which has been proven to be beneficial for deep convolutional neural networks using the ReLU activation function.

The last convolutional layer of the encoder is shaped so as to yield a (N, F)-shaped output. That is, the time and frequency dimensions are eliminated while leaving only the F latent space features extracted for each of the N input songs. This makes sense for spectrogram inputs, because the sonic qualities of a song that are useful for recommendation should be agnostic of the exact temporal and frequency makeup thereof.

4.4. Genre Predictor

The genre predictor network consists of 5 fully-connected layers that transform the F latent space features into the 8-dimensional output required to predict the genre. The genre predictor is kept relatively shallow compared to the encoder, because we want the encoder to produce a latent space with as much information as possible. Thus it is optimal to have most of the general compression learning happen in the encoder, while the predictor is only concerned with a few genre-specific transformations.

Layer	Input dim	Output dim
FC+ReLU	32	256
Conv+ReLU	256	128
Conv+ReLU	128	64
Conv+ReLU	64	8
Softmax	8	8

Table 3: Structure of the genre predictor

The output of the genre predictor is a softmax vector of 8

dimensions, corresponding to the categorical probability of a given spectrogram belonging to the 8 genres. The whole network is trained jointly via cross-entropy loss with respect to the correct genre classes.

5. Results

5.1. Genre Classification

Our best model was able to achieve near state-of-the-art genre classification accuracy of 51.4% using 2-dimensional convolution and no dropout or batch normalization. Previous state-of-the-art performance without data augmentation was 51.2%, achieved by Lee (2019) [12]. With data augmentation, Bian (2019) achieves state-of-the-art performance accuracy of 68.9% [2].

Model	Accuracy (%)
Transfer learning CNN (Lee, 2019)	51.2
Ours (convolutional autoencoder)	51.4
CNN w/ data aug. (Bian, 2019)	64.7
ResNet w/ data aug. (Bian, 2019)	66.7
DenseNet w/ data aug. (Bian, 2019)	68.9

Table 4: Our model approaches state-of-the-art performance while also producing compressed encoding vector

Considering the added difficulty of compressing the input into single encoding vector, it is expected that our model's accuracy would suffer slightly as compared to conventional models with no such restrictions. Despite this, our model was able to achieve impressive performance, and even beat out Lee (2019)'s previous state-of-the-art model. Bian (2019) clearly remains dominant, although the use of data augmentation makes it difficult to compare, as it effectively provides the model with additional training data.

An unexpected result was that batch normalization had an adverse effect on our model (Figure 3, right). Batch normalization is traditionally an essential tool for training deep neural networks, but in our case it actually hindered learning. We believe this may be due to the unique nature of our encoder architecture, but it may also be that music's asymmetrical relationship with time and frequency dimensions means that traditional methods found to work on images do not necessarily apply. Alternatively, it is possible that for the widely-varying spectrogram inputs, decreasing covariate shift during training with batch normalization actually increases the population difference between the train and test distributions, which hinders performance. A third possibility is that our batch size was simply too small to compute good batch statistics.

Dropout similarly failed to enhance performance, but did not hinder it, indicating that the network generalizes well. Dropout did help prevent deterioration of performance in long training sessions (100+ epochs, not pictured), but did not result in increased genre classification accuracy.

Our best performance was found using 2-dimensional convolution, which was also unexpected. Previous works have argued that 2-dimensional convolution over frequency and time is uninterpretable [16], and 1-dimensional convolutional filters have fewer parameters and are thus more computationally efficient as well. 2-d convolution was used by Choi (2016) to achieve state-of-the-art classification accuracy at the time [3], but many subsequent works have found best performance with 1-d convolution [2] [13].

This is likely due to the smaller 1-d filters being exposed to more input per parameter, and thus achieving better performance through increased weight sharing. However, our results prove that with an amenable architecture, 2-dimensional convolutions are able to capture more information about spectrogram patterns than 1-dimensional convolutions (Figure 3, left).

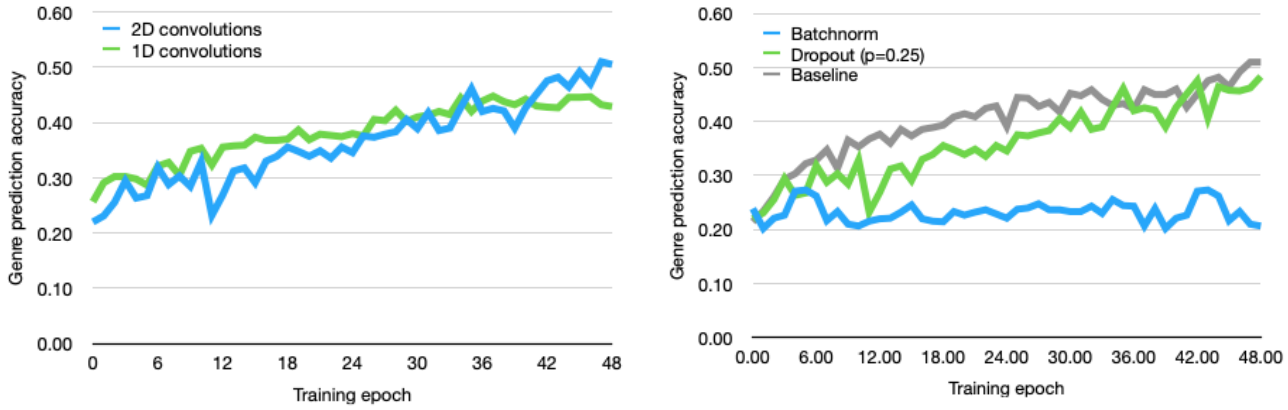

Figure 3: Left: 2D convolution over both frequency and time performs better than 1D convolution over just time. Right: batch normalization severely limits learning

5.2. Recommendation

We found that qualitatively, the nearest neighbors in our compressed latent space to a given song exhibited sonic similarities, usually in the form of percussion. This can be explained by the fact that percussion is one of the most consistent and prominent features in a given song. Compared to songs randomly chosen from the same genre, we found that our recommendations seemed to exhibit more similarities. This validates our approach, and is promising for our final goal of song recommendation and playlist creation.

Playlist creation resulted in somewhat less cohesive playlists than desired. While each song was similar to the original song, finding near neighbors did not guarantee that this similarity was in the same dimension. Thus, recommended songs sometimes differed from each other and sounded out of place. We suspect that this method of picking nearest neighbors in the latent space may not be the best way to generate cohesive playlists, as a user usually wants playlists to have some common attribute such as tempo or key.

However, as our model was trained strictly on genre, clusters in the resulting latent space were predominantly comprised of songs from the same genre. A promising direction for future work is the creation of playlists based on similarity in only a few latent space dimensions while ignoring the rest, a sort of playlist "signature", with the signature being dynamically selected by a user-driven algorithm.

6. Conclusion and Future Work

Our model was able to achieve near state-of-the-art performance on genre classification, while also performing the added task of explicitly compressing input audio data into an extremely low-dimensional latent space. This shows that it is possible to retain relevant musical information in a low-dimensional vector, to be later used for recommendation. The lack of standard evaluation metrics for recommendation is a lingering problem, but we are satisfied with our results using genre classification as a proxy metric.

6.1. Autoencoding

Our original architecture plan was to utilize a two-headed network approach, splitting after the encoder section: one head being the genre-classifier, and the other being an 'decoder' head. The decoder head is comprised of deconvolutional layers, which decompress the latent representation back into the original input dimensions. Unfortunately, this turned out to be extremely difficult to train and we were forced to abandon the idea. However, given more time and computing power, it may be possible to train such an architecture. This method would train the network to compress input data in such a way that the most information can be recovered, approaching a lossless compression.

We believe this would make latent representations more representative, and thus lead to better recommendations. Given no standard evaluation method for recommendations, however, it would be difficult to test this hypothesis and measure results quantitatively.

6.2. EchoNest Features

Another idea we investigated was training not only on genre, but also including musical features provided by EchoNest (now acquired by Spotify) such as energy, tempo, acousticness, key, etc. The Million Song Dataset (MSD) has many of such features, but unfortunately this dataset has not been maintained since its creation in 2010, and is no longer readily available to the public. The FMA includes some EchoNest features, but we had trouble with convergence and we suspect the high level features such as danceability were too abstract for the network to directly derive from the raw audio from scratch.

However, we hypothesize that starting from pre-trained networks on genre recognition, it may be plausible to perform transfer learning and train on the EchoNest features. We planned to evaluate this method using deconvolution with the decoder head, by comparing autoencoder performance when training on only genre versus genre and EchoNest. There is no clear way to evaluate the difference between the two quantitatively without this head.

6.3. Data Augmentation

In the future, we would like to investigate the use of data augmentation, as used by Bian (2019) [2]. This work found pitch shifting and overlapping time windows provides extra training data for the network to learn on, and led to state-of-the-art accuracy of 64.7%. However, Ulyanov argues for the mel-frequency dimension to be interpreted as channels instead, which is used successfully to achieve key-invariance. This may lead to better generalization and prevent the need for pitch shifting augmentation [16].

6.4. Residual Networks

We would also like investigate other network models such as ResNet and DenseNet, both of which were used by Bian (2019) in order to increase accuracy to 66.7% and 68.9%, respectively. However, since the field of audio processing is so different from traditional computer vision tasks, perhaps more specialized models can be created to account for the unique structure of music.

6.5. Recurrent Networks

Another area for future investigation is the use of recurrent neural networks (RNNs) for this application of music genre recognition and recommendation. Elementary work has been done exploring this method [8], but there is much to be tested. RNNs can also be used for sequence-aware

music recommendation, which is a natural extension of next-song recommendation.

References

[1] A. Berenzweig and B. Logan. A large-scale evaluation of acoustic and subjective music similarity measures. *Computer Music Journal*, 28:63–76, 2004.

[2] W. Bian and J. Wang. Audio-based music classification with densenet and data augmentation. 2019.

[3] K. Choi and K. Cho. Transfer learning for music classification and regression tasks. *18th International Society for Music Information Retrieval Conference*, 2017.

[4] S. Deng. Exploring user emotion in microblogs for music recommendation. *Expert Systems with Applications*, 42:9284–9293, 2015.

[5] S. Dieleman and B. Schrauwen. Deep content-based music recommendation. *International Conference on Neural Information Processing Systems (NIPS)*, 1:2643–2651, 2013.

[6] S. Dieleman and B. Schrauwen. End-to-end learning for music audio. *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1:6964–6968, 2014.

[7] A. Elbir and N. Aydin. Music genre classification and music recommendation by using deep learning. *Electronics Letters*, 56(12):627–629, 2020.

[8] G. Gessle and S. Åkesson. A comparative analysis of cnn and lstm for music genre classification. 2019.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, page 1026–1034, USA, 2015. IEEE Computer Society.

[10] A. Jiménez and F. José. Music genre recognition with deep neural networks.

[11] J. Kim and J. Urbano. One deep music representation to rule them all? a comparative analysis of different representation learning strategies. *Neural Comput. & Applic.*, 32:1067–1093, 2020.

[12] J. Lee and K. Choi. Deep learning for audio-based music classification and tagging. *IEEE signal processing magazine*, pages 41–51, 2019.

[13] J. Lee and K. Lee. Deep content-user embedding model for music recommendation. *Proceedings of DLRS 2018*, pages 01–05, 2018.

[14] T. Li and A. Chan. Automatic musical pattern feature extraction using convolutional neural network. *International Conference Data Mining and Applications*, 2010.

[15] M. Schedl. Deep learning in music recommendation systems. 2019.

[16] D. Ulyanov and V. Lebedev. Audio texture synthesis and style transfer. 2016.

[17] S. Zhang and L. Yao. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, 1:01–35, 2018.