Music Genre Recognition using Deep Learning

Asgharivaskasi, Arash Givehchian, Hadi Kasarla, Abhilash Shahsavari, Sina
Venkatswammy, Kishore

Abstract

Music genre recognition has been successfully explored over the last years and recently there has been increasing interest in applying deep learning approaches to tackle this problem. However, most of the proposed architecture are suited only for short music tracks and does not work well for longer tracks. In this paper, we propose a solution to mitigate this problem by introducing a segmentation algorithm which can extract useful information in a long audio sequences. We also introduce a novel modification to a RNN layer that can achieve the effect of regularization. We explore transfer learning, segmentation and RNN based approaches to music classification.

1. Introduction

Extracting Information from music is an interesting area of research from both machine learning and signal processing perspectives. Historically, there have been a lot of works using classical machine learning techniques like KNN [41][42][28], SVM [22][1][27], n-gram model [22][13][23], HMM [7], etc. to tackle this problem. However, none of them yielded a high accuracy for genre classification.

In the past few years, due to the advances in deep learning, there has been significant progress on retrieving information from a piece of music, and music genre classification has garnered good attention. Many related works take Mel-Spectrogram or short time Fourier transform (STFT) of the music track to create a 2D representation and propose Convolutional Neural Network (CNN) architectures to perform classification on it. [16][33][45][18][44][9][14]. Liu et al. [29] designed a CNN called Bottom-up Broadcast Neural Network (BBNN) which is equipped with a novel Broadcast Module (BM) which consists of Inception blocks and connects them by dense connectivity. They have gotten the best result among CNN-based architectures with a 93.9% accuracy on GTZAN dataset [40] and 96.7% accuracy on Ballroom dataset [6].

Recurrent Neural Network (RNN), a type of Neural Network, is widely used for processing sequential data and has the ability to learn long-term relationship. Among RNN architectures, Long Short-term Memory (LSTM) [24] and Gated Recurrent Unit (GRU) [8] are most suited to learning long-term patterns, as they overcome some problems like vanishing gradient, poor performance on long sequences faced by vanilla RNN. Some works [25][19][39] propose RNN architectures for the music genre classification task since music is inherently sequential. Wu et al. [43] introduce Independent Recurrent Neural Network (IndRNN) which can learn long-term relationship better than LSTM or GRU. The accuracy of their model on GTZAN dataset is 96% which is the current state-of-the-art result. Choi et al. [10] propose a CRNN approach to take advantage of CNN for local feature extraction and RNN for temporal summarization of the extracted features.

Transfer learning is widely used in Computer Vision and NLP applications to achieve good results using less amount of data. Transfer learning refers to re-using parameters that are trained on particular task for a target task which is similar. Choi et al. [11] train a Convnet for music tagging and then transfer learn music classification and other regression tasks from it. Song et al. [38] pretrained their RNN on Magnatagatune dataset [26] and use GTZAN dataset as the target dataset of genre classification.

Our proposed solution for music genre classification uses a deep learning architecture by transfer learning VGGNet and ResNet (which are pre-trained for image classification) on mel-spectrogram of music tracks. We also explore using an RNN architecture for the classification, which takes the raw audio signals as input. Moreover, all previous works are designed for short music tracks (mostly about 20 seconds to 1 minute) and they show poor performance on long music tracks. Training deep neural networks on long music tracks is challenging due to extremely long sequence length for raw audio. We will introduce a spectral clustering based segmentation algorithm (a non-linear embedding learning algorithm) to mitigate this problem.Here we extract the most important and representative segment of the music in an unsupervised manner and train our proposed network on these segments, instead of the whole music.

Mathematical problem formulation is provided in section 2. In section 3, we mention 3 datasets used to evaluate our

approaches. In section 4, we elaborate upon our proposed deep learning approaches for classification. There we also introduce skip-RNN which can be used to effectively regularize an RNN based model. In order to distinguish important parts of a song to be able to deal with long music tracks, we will propose a novel segmentation algorithm in section 5. Finally, we have evaluated the performance of our methods in section 6.

2. Problem Formulation

Music Genre Classification problem is the cornerstone of the research for Music Information Retrieval (MIR), in which the goal is to extract different information about a piece of music. Music Genre Classification is the task of assigning a specific genre to a piece of music. The goal is to take a raw music track as input and assign a genre (from a set of genres) as the label. In the training phase, we have training examples $x^{(1)}, x^{(2)}, x^{(3)}, ..., x^{(n)}$ from the space X along with their labels $y^{(1)}, y^{(2)}, y^{(3)}, ..., y^{(n)}$ where $y^{(i)} \in G$ and G is the set of music genres. In the training phase, the goal is to find a mapping $h : X \rightarrow G$ that assigns a label $h(x^{(i)})$ to each example $x^{(i)}$. In the test phase, for each test example x, we will predict $h(x)$ as its genre.

3. Datasets

Three datasets are used for evaluation of our methods: GTZAN, Extended Ballroom ,and 1517 Artists dataset. The GTZAN dataset has been widely used as a benchmarks for music genre classification by many researchers. Since this dataset is small, using The Extended Ballroom (augmented version of famous Ballroom dataset) and 1517 Artists datasets addresses this issue.

The GTZAN dataset which was collected and proposed by Tzanetakis et al. in [40], is used as target dataset in this paper. It has 10 genres and the input is 22050Hz, 16-bit, single channel WAV. The classes are balanced. The detail of GTZAN dataset is described in Table 1.

The Extended Ballroom was proposed in 2016 by Marchand et al. in [31], which extended the original Ballroom dataset [6]. The extended version contains 4180 audio tracks (6x larger than the original and has better audio quality). This dataset has imbalanced classes. We show its distribution in Table 1. The datasets are split into train and validation sets in 70/30 proportions.

The 1517 Artists collection which was introduced by Klaus Seyerlehner et al. in [34] is the largest among the three. It consists of 3100 freely available songs from 1517 artists. Each song is assigned to one of the 19 genres. Since the audio tracks in this dataset have variable lengths we have split each of them to 30 seconds tracks to make them compatible with our algorithms, resulting in our dataset to have 23000 audio tracks. The specific genres and the corresponding number of each genre are listed in Table 1.

GTZAN		1517 Artists		Extended Ballroom	
Genre	Tracks	Genre	Tracks	Genre	Tracks
Classical	100	Alternative & Punk	182	Cha Cha	455
Jazz	100	Blues	186	Slow Waltz	65
Blues	100	Childrens	164	Viennese Waltz	252
Metal	100	Classical	125	Tango	464
Pop	100	Comedy & Spoken Word	134	Samba	468
Rock	100	Country	187	Rumba	470
Country	100	Easy Listening & Vocals	175	Quickstep	497
Disco	100	Electronic & Dance	164	Jive	350
Hip Hop	100	Folk	185	Waltz	529
Reggae	100	Hip Hop	155	Salsa	47
		Jazz	177	Foxtrot	507
		Latin	163	Pasodoble	53
		New Age	175	Wcswing	23
		R&B & Soul	175		
		Reggae	172		
		Religious	172		
		Rock & Pop	181		
		Soundtracks & More	150		
		World	158		
Total	1000	Total	3100	Total	4180

Table 1. Datasets Discription

Algorithms	full songs' test accuracy	segmented songs' test accuracy
SVM	43.56%	**45.67%**
Soft-Max	38%	42.67%
Decision Tree	**50.08%**	39.64%
Naive Bayes	30.67%	30.67%
KNN	27.67%	34%

Table 2. Evaluation of some classic algorithms on GTZAN

4. Experiments

4.1. Classic Machine Learning Approaches

We have evaluated some popular machine learning algorithms as a baseline to compare our approaches. For the sake of being fair, we have evaluated these algorithms on both full songs (30 seconds) and 3-second segments of the songs (followed by voting) on GTZAN dataset. The results are provided in Table 2, among which the Decision Tree model gave the best result.

4.2. Features Preparation

We used the spectrogram representations of a song as our input features for our models. A spectrogram is a 2D representation of a signal, with time (x-axis) and frequency (y-axis). A colormap is used to quantify the magnitude of a given frequency within a given time window. For our experiments, each audio signal was converted to MEL spectrogram (MEL frequency bins on the y-axis). Parameters that we used to generate the power spectrogram using STFT: sampling rate = 22050, frame/window size = 2048, hop size

= 1024 (50% overlap), number of mel bins = 128. Sample spectrogram of each class is shown in Figure 1.

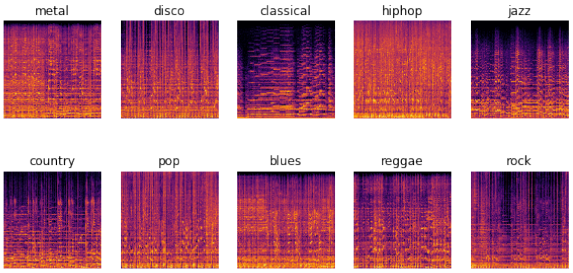

Figure 1. Sample spectrograms from each music genre

4.3. Transfer learning Approach

Our main idea is to interpret the 2D spectrogram as an image and assume it has all the characteristic patterns required to differentiate between genres. Hence we use these spectrograms as input to train our CNN, which has shown good performance on image classification tasks. In this paper, a CNN architecture known as VGG-16, which was the top performing model in the ImageNet Challenge 2014 [35] was used. We discard the Fully connected(FC) layers in the VGG-16 and add our own layers. The weights, which were pretrained on ImageNet [17], were fine-tuned to the spectrograms of the audio. After experimenting with the sizes of the FC layers, we noticed that the sequence of sizes 256-128-64 gives the best results. It is interesting to note that learning from the domain of images can be transferred to classify audio signals represented in 2D and produce good results.

In a similar fashion, we train our model using a pretrained ResNet architecture [21]. We incorporate recent advances in Deep Learning that enabled us to perform better than existing approaches with the same data and model architectures. The techniques used include cyclic learning rate [36], 1cycle policy [37], use of small batch sizes for regular learning [20] and use of large batch size for cyclic learning rate [37]. The ResNet architecture learns better representations of data than a VGGNet and this fact is reflected in the improved accuracy we got using the ResNet model.

In this work, we also introduce a novel way to perform model regularization by adding skip-connections to the hidden state of the RNN (GRU). We refer to it as skip-RNN and the results are detailed in the evaluation section.

4.4. RNN

One concern with the use of transformed input (like the MFCCs used to transform 1D signals to 2D images) is that there is a loss of information. Also, we can learn a better representation using autoencoders [4] or other algorithms

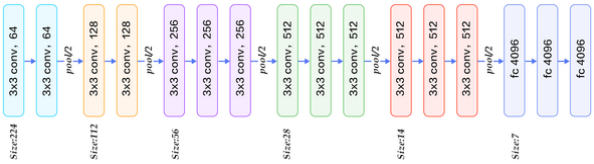

Figure 2. VGG-16 architecture

for obtaining a compressed representation of the input. We trained an RNN with GRU (Gated Recurrent Unit) cells and two fully connected layers on top of it. The RNN is trained from scratch. This model had two forms of input - the MFCCs and the raw audio input. The raw audio is 30 second long and the content for each second is passed as input for one timestep of the RNN. This model does not perform as well as the transfer learning approach due to the reason that the datasets are small and the model is trained from scratch.

5. Segmentation

One of the major challenges of music genre classification tasks is to extract the genre-related part of a soundtrack in order to improve the learning accuracy. In other words, songs usually contain parts which are not relevant to their representative genre, and 'distract' the learning algorithm to properly model the feature distribution of each genre. This is one of the reasons that it is commonplace in MIR literature to use song excerpts instead of the full soundtrack. Here, we propose an unsupervised method to automatically extract the most relevant part of an input song as a means to train the classifier on original music pieces without the requirement of manual excerpting. Inspired by non-linear manifold learning algorithms, we aim to map input soundtrack into a high-dimensional space, and learn the principal component of the soundtrack, where we believe the most genre-related information can be extracted from. To reach such a goal, we present a three-step algorithm as follows:

5.1. Time-Domain to Graph Mapping

A classical problem in signal processing is to determine statistically stationary segments in a time signal. This problem, which is commonly known as adaptive segmentation, can be quantitatively formulated as a nested hypothesis testing framework; where we assume that the whole signal is stationary until otherwise is proven. Specifically, we use the spectral error measure (SEM), proposed by [5], as a feature extraction method in order to quantify non-stationarities in time-series. The fundamental idea of this method is the presumption that the characteristic behavior of a small segment of music signal can be satisfactorily modeled by an auto-

regressive (AR) process which is defined as

$$y[n] = \sum_{k=1}^{p} a_k y[n-k] + u[n] \tag{1}$$

where $y[n]$ is a function of time, $u[n]$ is white Gaussian noise with variance σ^2, and a_k is the k-th coefficient of the p-order AR model. An AR model can be seen in various ways; essentially, it is a predictive model based on the previous samples of the time series. However, in the frequency domain, an AR model can also be interpreted as a parametric estimation of the power spectrum density of a time series, which is more interesting for our application. As in [5], SEM between two time windows (reference and sliding windows) is obtained as shown in Algorithm 1. The intuition behind the SEM method is the fact that if the AR estimation of power spectra of both windows were identical, the residuals from the sliding window would be white noise; therefore, SEM essentially calculates the power spectral deviation of the residuals from white noise with variance $\hat{\sigma}^2_{ref}$.

Algorithm 1 Spectral Error Measure between Two Time Windows [5]

Input: Reference window $w_{ref}[n]$, Sliding window $w_{sld}[n]$, Window length M, Order of AR process p

1: Estimate an p-order AR process as in (1) with coefficients $\hat{W}^{-1}_{ref}(z)$ and noise power $\hat{\sigma}^2_{ref}$ from $w_{ref}[n]$.
2: Pass $w_{sld}[n]$ through the inverse filter $\hat{W}_{ref}(z)$ to obtain residual signal $s_{res}[n]$.
3: Calculate estimated auto-correlation function $r_s[m]$ of the residual signal $s_{res}[n]$.
4: Obtain:

$$SEM_{w_{ref}[n],w_{sld}[n]} = \left(\frac{\hat{\sigma}^2_{ref}}{r_s[0]} - 1\right)^2 + 2\sum_{k=1}^{M}\left(\frac{r_s[k]}{r_s[0]}\right)^2 \tag{2}$$

We make use of SEM as a distance metric in order to build a graph that represents an input soundtrack.

First, we divide the signal into N_w fixed length windows during which it is assumed that the signal is stationary. Now, in order to quantify non-stationarity between a pair of windows, we calculate SEM of the two. However, for a pair of windows (i th window and j th window for example), the SEM value is not symmetric. To tackle this problem, as proposed by [3], we assign the average SEM value for each pair of windows; i.e., we first obtain SEM considering i-th and j-th windows as the reference and sliding windows, respectively, then we substitute the indices of reference and sliding windows and calculate SEM once again and, finally, take the average SEM value as the metric of statistical distance between these two time windows.

In order to formulate the time-variant statistical structure of a soundtrack, we build an undirected graph $\mathcal{G}$ with N_w vertices where each vertex represents a time window and weight of the connection between each pair of vertices is defined as a non-linear function of SEM between the two corresponding windows. Algorithm 2 describes building graph $\mathcal{G}$ as a mapping from the time domain to graph vertices.

Algorithm 2 Time-series to Graph Transformation

Input: Time domain signal $f[n]$, Number of windows N_w

1: Segment $f[n]$ into N_w fixed length time windows. The windows can be either overlapping or non-overlapping.
2: Construct statistical distance matrix $\mathbf{S} \in \mathbb{R}^{N_w \times N_w}$ as:

$$(\mathbf{S})_{i,j} = SEM_{w_i[n],w_j[n]} \tag{3}$$

where $1 \leq i, j \leq N_w$, and $w_i[n]$ and $w_j[n]$ are i-th and j-th segments of $f[n]$, respectively.
3: Symmetrize $\mathbf{S}$ as:

$$\mathbf{S}_{sym} = \frac{\mathbf{S} + \mathbf{S}^T}{2} \tag{4}$$

4: Construct adjacency matrix $\mathbf{A} \in \mathbb{R}^{N_w \times N_w}$ of graph $\mathcal{G}$ as:

$$(\mathbf{A})_{i,j} = \begin{cases} exp(-(\mathbf{S}_{sym})^2_{i,j}) & i \neq j \\ 0, & i = j \end{cases} \tag{5}$$

for $1 \leq i, j \leq N_w$
5: Build $\mathcal{G}(V, E)$ as an undirected graph with vertex set V and edge set E where V_i represents the i-th segment of time-series $f[n]$ and $E_{i,j} = (\mathbf{A})_{i,j}$.

In the first step, the number of windows N_w should be selected with regard to the trade-off between the assumption of stationarity during each window and the accuracy of AR process estimation. Considering the problem of SEM value asymmetry for a pair of windows, the obtained distance matrix in (3) is not necessarily symmetric. To resolve this, we assign the average SEM value for each pair of windows i.e., we symmetrize distance matrix S as in (4). Step 4 describes the transformation from a distance matrix to an adjacency (similarity) matrix, emphasizing that there is no self-connecting edge in the corresponding graph. Now with the resulting adjacency matrix, we can identify the time windows with similar statistical behavior. In other words, we try to extract clusters in the graph $\mathcal{G}$ with the corresponding adjacency matrix $\mathbf{A}$. Figure 3, 4 and 5 depict the mapping process for an example soundtrack.

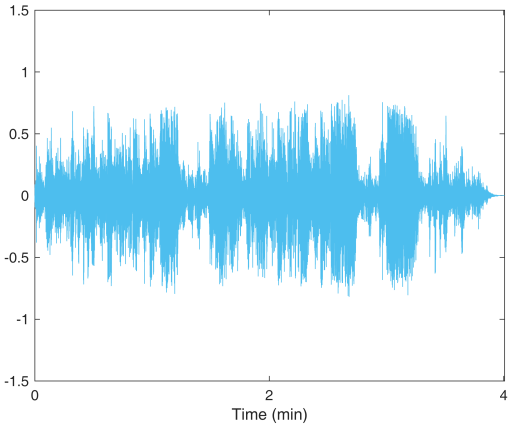

Figure 3. Raw audio file

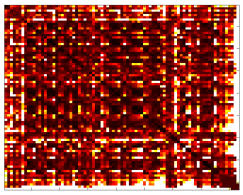

Figure 4. Statistical distance matrix $\mathbf{S}$ obtained from the example raw audio. Bright pixels show higher matrix values.

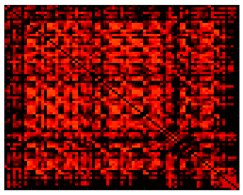

Figure 5. Resulting graph adjacency matrix $\mathbf{A}$. Bright pixels show higher matrix values.

5.2. Graph Clustering

From a graph theory aspect, one approach to detect communities in a network is solving the Min-Cut problem. However, in practice, it does not lead to satisfactorily large clusters [30]. One can deal with this problem by introducing balancing conditions to the Min-Cut problem, which is exactly what the N-cut and Ratio-Cut algorithms do. Unfortunately, by presenting the balancing condition to the problem, the previously easy to solve Min-Cut problem becomes NP-hard [30]. One family of efficient solutions to relaxed versions of the N-cut and Ratio-Cut problems are Spectral Clustering algorithms, which can be solved by standard linear algebra methods. Particularly, using a relaxed version of the N-cut problem, we aim at clustering the graph $\mathcal{G}$ with corresponding adjacency matrix $\mathbf{A}$ by normalized spectral clustering. The most important instrument for spectral clustering is a graph Laplacian matrix. It is well-studied in the field of spectral clustering that the eigenvectors and eigenvalues of the Laplacian matrix are a natural index of connectivity and partition of the representing graph [32, 2, 15, 12]. Therefore, one can identify node communities by simply performing eigenvector decomposition (EVD) of Laplacian matrix $\mathbf{L}$. As described in [32], the (normalized) Laplacian matrix is constructed as:

$$\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \tag{6}$$

where $\mathbf{D} \in \mathbb{R}^{N_w \times N_w}$ is a diagonal degree matrix whose (i,i)-element is the sum of $\mathbf{A}$'s i th-row:

$$(\mathbf{D})_{i,i} = \sum_{j=1}^{N_w} (\mathbf{A})_{i,j}. \tag{7}$$

For a symmetric adjacency matrix $\mathbf{A}$ with non-negative elements, the two key properties of the corresponding Laplacian matrix $\mathbf{L}$ are [30, 32]:

(1) $\mathbf{L}$ has N_w real-valued eigenvalues $\lambda_{N_w} \leq \cdots \leq \lambda_1 = 1$.

(2) The multiplicity k of eigenvalue 1 of matrix $\mathbf{L}$ equals the number of connected components in the graph $\mathcal{G}$. The corresponding k eigenvectors carry the information of which connected component each vertex belongs to.

It is important to mention that in most practical cases the network graph is fully connected. Hence, there is no more than one connected component; yet there are several communities in the network that we are interested to identify. In these cases, the property (2) can be restated as saying that the multiplicity k of eigenvalues of $\mathbf{L}$ distinctively close to 1 carries the information about the number groups in the network [30]. In order to algorithmically find clusters in graph $\mathcal{G}$ for such settings, as proposed by [32], one should follow the steps indicated in Algorithm 3.

As one may notice, the main idea behind the algorithm is based on the EVD of a similarity measure, which resembles Principal Component Analysis (PCA). In fact, it has been shown that spectral clustering can be viewed as PCA over a graph where there is no constraint on the distribution of data in the feature space [30, 32]. Hence, spectral clustering can be used as a means to learn embeddings over a non-linear manifold in higher-dimensions. Practically, we mapped the soundtrack signal from the time domain into a high-dimension manifold and then tried to learn the main components of the manifold as the connecting components of the underlying graph in an unsupervised manner. Figure 6 shows the spectral clustering on the graph built upon our example soundtrack. In the next step, we apply the inverse mapping from the high-dimensional graph space into the time domain to obtain the temporal manifestations of the graph components.

Algorithm 3 Normalized Spectral Clustering [32]

Input: Graph $\mathcal{G}(V, E)$ with adjacency matrix $\mathbf{A} \in \mathbb{R}^{N_w \times N_w}$, Number of clusters k

1: Form the normalized Laplacian matrix $\mathbf{L} \in \mathbb{R}^{N_w \times N_w}$ of graph $\mathcal{G}$ as (6).

2: Find $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k$, the k largest eigenvectors of $\mathbf{L}$ (chosen to be orthogonal to each other in the case of repeated eigenvalues), and form the matrix $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k] \in \mathbb{R}^{N_w \times k}$. The parameter k is determined with the prior knowledge of number of clusters in the graph.

3: Form the matrix $\mathbf{T} \in \mathbb{R}^{N_w \times k}$ from $\mathbf{U}$ by normalizing each row of $\mathbf{U}$ to have unit length, i.e.:

$$(\mathbf{T})_{i,j} = (\mathbf{U})_{i,j} / \Big(\sum_{j=1}^{k} (\mathbf{U})_{i,j}{}^2 \Big)^{1/2} \tag{8}$$

4: Treating each row of $\mathbf{T}$ as a data point in $\mathbb{R}^k$, cluster them into k subsets using k-means or any other clustering algorithm.

5: Assign the label y_i of the original vertex V_i to j if row i of the matrix $\mathbf{T}$ was assigned to cluster j.

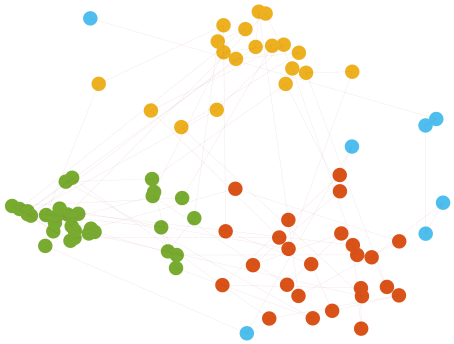

Figure 6. Graph representation of the example soundtrack. Vertices are localized using multidimensional scaling (MDS) as a dimension reduction method in order to visualize the high-dimensional distribution of vertices in 2D. Connecting lines indicate the temporal ordering of the corresponding time windows in the soundtrack. Vertices with the same color indicate members of one cluster. Notice that since we analyzed the graph in the high-dimensional space, some far away points in 2D have fallen into the same cluster (e.g. blue vertices).

5.3. Graph Space to Time Domain Inverse Mapping

Given the clustered graph of input soundtrack, we assign each time window its corresponding label according to the output of the previous step. Algorithm 4 mathematically expresses the inverse mapping step. Figure 7 shows the final segmentation results for our sample soundtrack.

Algorithm 4 Graph to Time Domain Inverse Mapping

Input: Time domain signal $f[n]$, Number of windows N_w, Graph $\mathcal{G}(V, E)$ with vertex label $\mathbf{y} \in \mathbb{N}^{N_w}$ ($y_i \in \{1, 2, ..., k\}$)

1: $l = length(f[n])/N_w$

2: For $i = 1 : N_w$

$y_i \rightarrow f[(i-1) \times l + 1 : i \times l]$

Now we have the whole soundtrack segmented into different parts. Here, we can simply pass original full-length music pieces to the deep learning algorithm with the segmentation block attached before the first layer of the network in order to be able to train the algorithm on full-length soundtracks instead of 30-second excerpts. In the next section, we will study the effect of the unsupervised segmentation on the supervised deep learning algorithm.

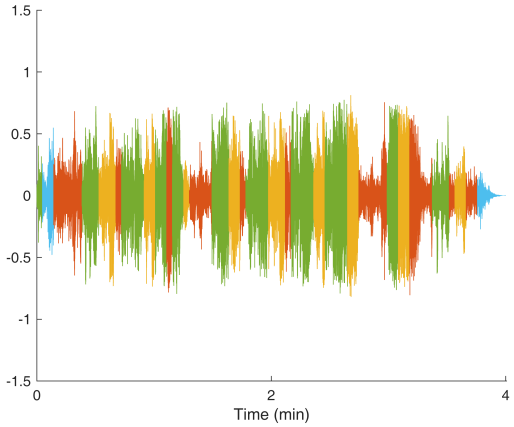

Figure 7. Segmentation results for the example soundtrack. Segments with the same color indicate members of one cluster. It is easy to observe that time windows with the same cluster almost repeat in time, which is an usual pattern in music pieces.

6. Evaluation

Figure 8 shows the training of VGG16 on GTZAN dataset using the full spectrum for each song. Beyond 30 epochs, the model overfits. We used early stopping to save the best model which had an accuracy of **75%**. We tried splitting the songs into 3 second segments sequentially (different from the segmentation mentioned in previous section) and train on these segments. The result of this training is shown in Figure 10, which are comparable with the results of training on full song spectrum.

Next we tried to incorporate our Segmentation algorithm results and learn only using the largest cluster found in the

song (Figure 11). It is interesting to note that using only one segment for each input (from the largest cluster) we get an accuracy of **67%**. We expect that this novel method of integrating the aforementioned segmentation algorithm and Deep Neural Network would work better if we had songs of longer length. In our case we had only 30 seconds songs and the genre related information was rich and uniformly distributed throughout. For longer songs our segmentation algorithm would perform even better in extracting the strong signal components from the audio, thereby improving the performance of our deep learning model. Figure 9 shows the training of VGG16 on Extended ballroom dataset. The best model gave an accuracy of 63.6%.

We trained a pretrained ResNet34 on GTZAN and Extended Ballroom datasets. This model gave us the best results. The validation accuracy for GTZAN dataset is **79%**. For the extended Ballroom we achieved an accuracy of **75.4%** on the validation set. The loss and accuracy plots for GTZAN and the Extended Ballroom datasets is shown in (Figure 12) and (Figure 13) respectively. We can notice that the loss/accuracy plots are cyclic - increasing and decreasing, with an effective downward/upward trend. This is one of the effects of cyclic learning rate which was used for these experiments. Training a GRU-RNN with a Fully Connected (FC) layer on top, we get validation accuracy of 40.3% for GTZAN dataset. Using a deeper network with Dropout regularization we do not get any improvements in accuracy as the dropout is added to the FC layers and we suspect that the RNN layer is the performance bottleneck. It is to note that the RNN was trained from scratch and as a result does not provide superior performance over the transfer learning approaches. The input was the MFCCs with each sample window as the time step. We get similar results for the Extended Ballroom dataset. These results are summarized in Table 3.

One recurring theme in training an RNN for these two datasets was that the train accuracy was very good, but the validation accuracy was poor, i.e. the models we overfitting to the train data. In our work we introduce skip-RNN. Skip-RNN is a regualar RNN (GRU here) cell where we add skip connections for the hidden state of the RNN layer which is propagated during each time step. The length of the skip (k) is a hyper parameter to be determined. The effect of a skip-RNN is that it leads to regularization of the model, without affecting the validation accuracy. The train and validation losses for GTZAN and Extended Ballroom using regular RNN and our skip-RNN is shown in (Figure 14) and (Figure 16) respectively. Similarly the accuracies are show in (Figure 15) and (Figure 17) respectively. We can see in the figures, that the train accuracy is less using our skip-RNN, but the validation accuracy is unaffected. Thus our skip-RNN lets us regularize the model. To the best of our knowledge, our work is the first to add skip connections to hidden layers of RNN and achieve the effect of regularization.

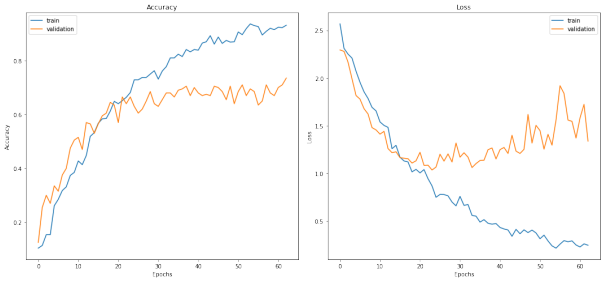

Figure 8. GTZAN dataset - VGG16 training

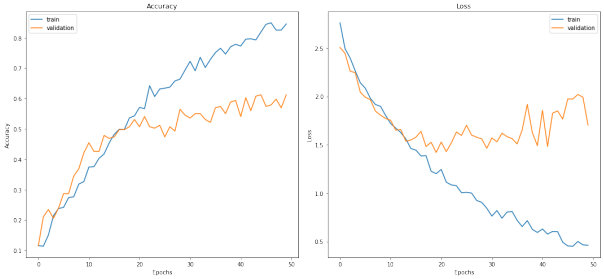

Figure 9. Extended ballroom - VGG16 training

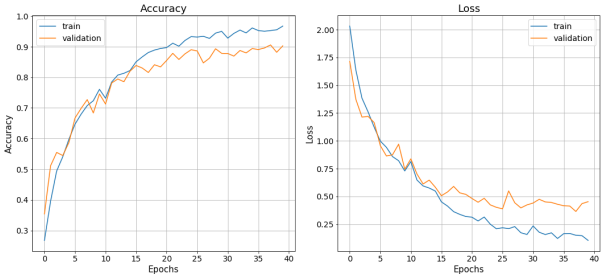

Figure 10. GTZAN using 3sec segments - VGG16 training

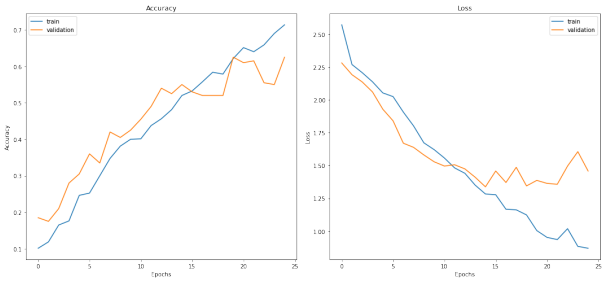

Figure 11. Using Segmentation results and training VGG-16

7. Future works

In this paper we experimented only with Mel-Spectrogram as the input features for our models. However,

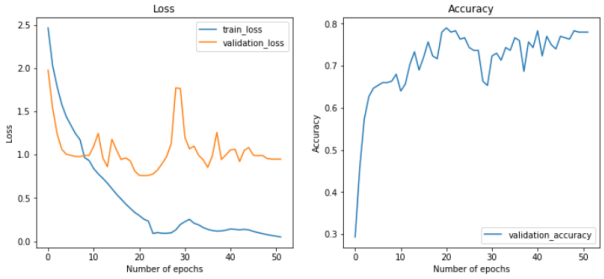

Figure 12. GTZAN dataset - ResNet34 loss and accuracy

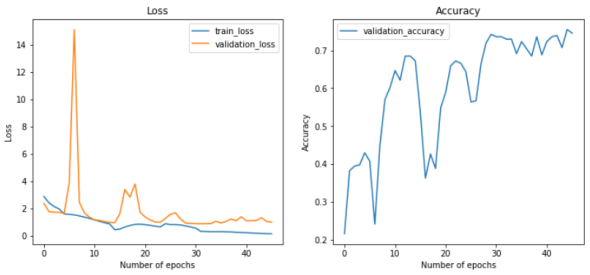

Figure 13. Extended Ballroom dataset - ResNet34 loss and accuracy

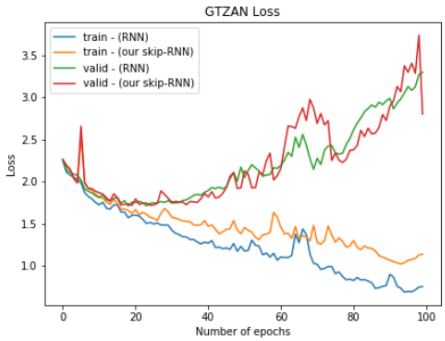

Figure 14. Loss plot on GTZAN for RNN and skip-RNN

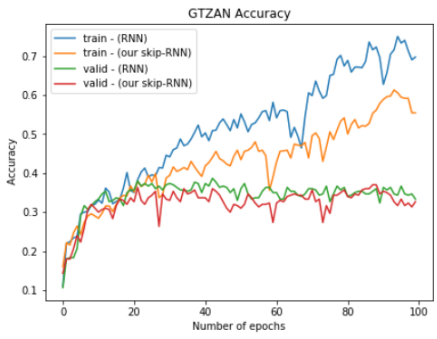

Figure 15. Accuracy plot on GTZAN for RNN and skip-RNN

there are several other frequency domain features which were proved to be effective like Spectral centroid, Chromagram, Spectral Contrast and Roll-off. The concatenation of such features could yield a good result using our proposed deep learning methods. Future work can explore effective

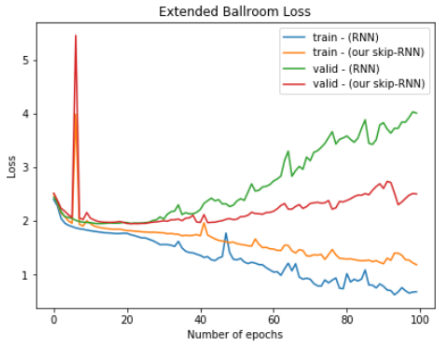

Figure 16. Loss plot on Extended Ballroom for RNN and skip-RNN

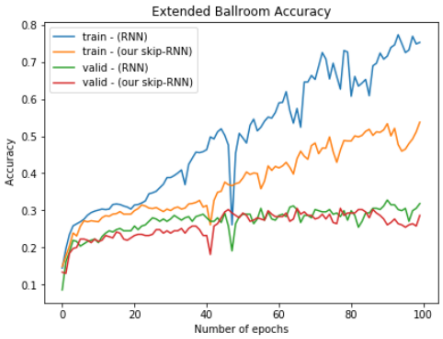

Figure 17. Accuracy plot on Extended Ballroom for RNN and skip-RNN

	GTZAN	Extended Ballroom	1517 Artists
VGG-16	75%	63.6%	62%
ResNet34	**79%**	**75.4%**	-

Table 3. Evaluation of Deep Learning Approaches

heuristics to select the skip length (k) for skip-RNN.

8. Conclusion

In this work we study music genre recognition. We experiment classical Machine Learning approaches to set a baseline for Deep Learning approaches. We show how a spectrogram of audio signal can be treated as an image and transfer the knowledge gained in the domain of images (transfer learn) to classify audio input. VGG16 and ResNet architectures (pretrained on ImageNet dataset) were used for the transfer learning. ResNet was found to perform better compared to VGG16. We also experimented an RNN based approach where we trained a GRU-RNN. As this model was trained from scratch and due to the size of the used datasets, we could not get significant results. We then introduce two novel approaches. The first one is using the results of the segmentation algorithm to train our Deep Learning models and shows promising results. In our second novel approach we introduce skip-RNN which enables regularization of RNN based models.

References

[1] J. Abeßer, H. M. Lukashevich, C. Dittmar, and G. Schuller. Genre classification using bass-related high-level features and playing styles. pages 453–458, 2009. 1

[2] C. Alpert. Spectral Partitioning: The More Eigenvectors, The Better. *32nd Design Automation Conference*, 1995. 5

[3] R. Aufrichtig, S. B. Pedersen, and P. Jennum. Adaptive Segmentation of EEG Signals. *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 1991. 4

[4] D. H. Ballard. Modular learning in neural networks. pages 279–284, 1987. 3

[5] G. Bodenstein and H. M. Praetorius. Feature Extraction from the Electroencephalogram by Adaptive Segmentation. *Proceedings of the IEEE*, 1977. 3, 4

[6] P. Cano, E. Gómez Gutiérrez, F. Gouyon, P. Herrera Boyer, M. Koppenberger, B. S. Ong, X. Serra, S. Streich, and N. Wack. Ismir 2004 audio description contest. 2006. 1, 2

[7] W. Chai and B. Vercoe. Folk music classification using hidden markov models. 6(6.4), 2001. 1

[8] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 1

[9] K. Choi, G. Fazekas, and M. Sandler. Explaining deep convolutional neural networks on music classification. *arXiv preprint arXiv:1607.02444*, 2016. 1

[10] K. Choi, G. Fazekas, M. Sandler, and K. Cho. Convolutional recurrent neural networks for music classification. pages 2392–2396, 2017. 1

[11] K. Choi, G. Fazekas, M. Sandler, and K. Cho. Transfer learning for music classification and regression tasks. *arXiv preprint arXiv:1703.09179*, 2017. 1

[12] F. Chung. Spectral Graph Theory. *CBMS Regional Conference Series in Mathematics, Conference Board of the Mathematical Sciences, Washington*, 1997. 5

[13] D. Conklin. Melodic analysis with segment classes. *Machine Learning*, 65(2-3):349–360, 2006. 1

[14] Y. M. Costa, L. S. Oliveira, and C. N. Silla Jr. An evaluation of convolutional neural networks for music classification using spectrograms. *Applied soft computing*, 52:28–38, 2017. 1

[15] N. Cristianini, J. Shawe-Taylor, and J. Kandola. Spectral kernel methods for clustering. *Advances in Neural Information Processing Systems*, 2002. 5

[16] J. Dai, W. Liu, C. Ni, L. Dong, and H. Yang. "multilingual" deep neural network for music genre classification. 2015. 1

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. 2009. 3

[18] M. Dong. Convolutional neural network achieves human-level accuracy in music genre classification. *arXiv preprint arXiv:1802.09697*, 2018. 1

[19] M. Freitag, S. Amiriparian, S. Pugachevskiy, N. Cummins, and B. Schuller. audeep: Unsupervised learning of representations from audio with deep recurrent neural networks. *The Journal of Machine Learning Research*, 18(1):6340–6344, 2017. 1

[20] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 3

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. pages 770–778, 2016. 3

[22] R. Hillewaere, B. Manderick, and D. Conklin. Global feature versus event models for folk song classification. 2009:10th, 2009. 1

[23] R. Hillewaere, B. Manderick, and D. Conklin. String methods for folk tune genre classification. 2012:13th, 2012. 1

[24] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1

[25] J. Jakubik. Evaluation of gated recurrent neural networks in music classification tasks. pages 27–37, 2017. 1

[26] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie. Evaluation of algorithms using games: The case of music tagging. pages 387–392, 2009. 1

[27] M. Li and R. Sleep. Improving melody classification by discriminant feature extraction and fusion. 2004. 1

[28] M. Li and R. Sleep. Melody classification using a similarity metric based on kolmogorov complexity. *Sound and Music Computing*, 2012, 2004. 1

[29] C. Liu, L. Feng, G. Liu, H. Wang, and S. Liu. Bottom-up broadcast neural network for music genre classification. *arXiv preprint arXiv:1901.08928*, 2019. 1

[30] U. V. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 2007. 5

[31] U. Marchand and G. Peeters. The extended ballroom dataset. 2016. 2

[32] A. Y. Ng, M. I. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 2001. 5, 6

[33] C. Senac, T. Pellegrini, F. Mouret, and J. Pinquier. Music feature maps with convolutional neural networks for music genre classification. page 19, 2017. 1

[34] K. Seyerlehner, G. Widmer, and T. Pohle. Fusing block-level features for music similarity estimation. pages 225–232, 2010. 2

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[36] L. N. Smith. Cyclical learning rates for training neural networks. pages 464–472, 2017. 3

[37] L. N. Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018. 3

[38] G. Song, Z. Wang, F. Han, and S. Ding. Transfer learning for music genre classification. pages 183–190, 2017. 1

[39] C. P. Tang, K. L. Chui, Y. K. Yu, Z. Zeng, and K. H. Wong. Music genre classification using a hierarchical long short term memory (lstm) model. 10828:108281B, 2018. 1

[40] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002. 1, 2

[41] P. van Kranenburg, A. Volk, and F. Wiering. A comparison between global and local features for computational classification of folk song melodies. *Journal of New Music Research*, 42(1):1–18, 2013. 1

[42] G. Velarde, T. Weyde, and D. Meredith. An approach to melodic segmentation and classification based on filtering with the haar-wavelet. *Journal of New Music Research*, 42(4):325–345, 2013. 1

[43] W. Wu, F. Han, G. Song, and Z. Wang. Music genre classification using independent recurrent neural network. pages 192–195, 2018. 1

[44] J. Yang. Music genre classification with neural networks: An examination of several impactful variables. 2018. 1

[45] W. Zhang, W. Lei, X. Xu, and X. Xing. Improved music genre classification with convolutional neural networks. pages 3304–3308, 2016. 1