Music Genre Classification using Mid-Level Features

Hannah Bollar
University of Pennsylvania
hbollar@seas.upenn.edu

Shivangi Misra
University of Pennsylvania
shivangi@seas.upenn.edu

Taylor Shelby
University of Pennsylvania
teshelby@seas.upenn.edu

Abstract

The goal of our project was to classify music genres using more human intuitive features. We used mid level features such as those found in sheet music, including pitch, rhythm, tempo, and instruments. For a large dataset, this information is obtainable from MusicXML files. In order to extract relevant patterns in temporal and spatial information from these features, long-short term memory (LSTM) recurrent neural networks were used for the classification task. Despite a messy data set and minimal features, we were able to classify 5 genres with an accuracy of approximately 63%.

1 Credits

Though we made our own initial derivations and explanations for how we will be implementing this project, we found both Bahuleyan's ***Music Genre Classification using Machine Learning Techniques*** and McKay's ***Automatic Genre Classification of MIDI Recordings*** to be incredibly helpful throughout this process (Bahuleyan, 2018; McKay, 2004).

2 Introduction

Music genre classification has been addressed by machine learning from a variety of perspectives and approaches (Z. Fu and Zhang, 1981). There is inherently difficulty in this classification due to the somewhat subjective nature of the definitions of genre, since the genres themselves have been developed over time based on a collective musical understanding instead of a defined set of prewritten rules. These characteristics are often difficult to pinpoint as they may be due to patterns in the rhythm of the song, instruments used to play the song, the lyrical content of the song, or a combination of these and more.

Since we do not clearly understand how humans classify genre, it is hard to select features to pass into various machine learning algorithms without understanding first how important or how much each of the features acts as a defining attribute of said genre. Much of the current research into music genre classification focuses on low level audio based features, such as amplitude modulation and zero crossing rate (Z. Fu and Zhang, 1981); however, there are many mid level features fundamental to the typical human approach to genre classification which are difficult to extract from frequency analysis.

3 Background

3.1 Music and Features

Features such as rhythm, harmony, and chord progression are key indicators, used directly by musicians and intuitively by non-experts, yet these are difficult to extract from audio spectrogram data. Though research is underway regarding better ways to extract these features from audio, it seems prudent to first demonstrate their effectiveness as identifiers (McKay, 2004). To do this, we propose extracting features including rhythm, harmony, and chord progression from MIDI data files.

It has been shown that using these mid level features has the potential to provide results comparable to low-level features, with a smaller feature space (C. Pérez-Sancho and Iñesta, 2009; M. G. Armentano and Cardoso, 2016); however, we believe there is more to be explored.

To work with these features, we extract data from music XML files. As there is no large database of these, we found a MIDI dataset and converted the files into XMLs using Musescore (Mus). MIDI's (Musical Instrument Digital Interface) formatting acts as a music storage and trans-

fer protocol which carries 16 channels of information, each containing event messages that dictate notation, pitch, velocity, vibrato, panning, or clock signals for tempo in songs. Music XML also includes this data, but in a readable text format, explicitly mentioning things such as the pitch and rhythm type of notes which make it ideal for extracting the mid level features we wish to utilize. This process is explained in detail in section 4.3.

3.2 Algorithms

Due to our disparate featurization, we spent some time deciding between Recurrent Neural Networks (RNNs) and Support Vector Machines (SVMs). The following is an explanation and pros and cons of each.

RNNs use internal states of the nodes to process information as well as the incoming stream of data. These nodes act as a layering system with a directed connection to every other node in the following layer. The nodes themselves have an activation state, weight, and tags for being input, output, or even hidden nodes depending on what layer in which they are placed.

One main problem with RNNs is that they often face "Vanishing Gradient" problem. That is, the update value returned from the partial derivative of the error function with respect to the current weight function is too small. This leads to the weight not changing enough or even noticeably at all (due to decimal storage limits); it can either heavily slow down training while stuck in this chasm or even completely stop the network from progressing entirely. To get around this issue to still use RNNs we consider the Long Short-Term Memory (LSTM) variation of RNNs for our work. This variation is beneficial since it uses feedback connections (instead of just feedfoward) to augment the error and update calculations.

SVMs use classification and regression analysis to analyze data. Using this technique, we would need to implement multiclass SVMs with a kernel trick for our output to converge as expected. These have been used to some success in other music classification machine learning research, so we believe them to be a viable alternative (M. I. Mandel, 2005). However, given that our feature set is different than the commonly used ones for music-classification, we opted not to use them for our project.

3.3 Thoughts and Expectations

Experiments using mid level features similar to those that we have used have resulted in an accuracy up to 85%; however, with our limited feature set, we expect to have similar accuracy to genre classification experiments using equally limited features of approximately 40%. (Z. Fu and Zhang, 1981).

Even with those expectations, we did foresee some issues. It is possible that the mid-level features we have selected on their own will not provide enough data (since we hand picked them based on usability and accessibility), and even that the way we have chosen to input these features to our network can generate differing results compared to other input representations. This further enhances the well established idea that that music genre classification is a complex problem, dependant on many features that even present scholars are still looking to understand (Z. Fu and Zhang, 1981).

Furthermore, due to the sequential nature of music chord progressions, we have chosen to both use RNNs and to encode some sequence into our features by utilizing multiple measures at a time. This is something from which we expected an interesting output, since we had not yet determined if using both of these is redundant or critical to capturing the sequential aspect of music.

4 Implementation Details

4.1 Genres

From a music theory standpoint, chord progressions are a key factor in distinguishing between genres, so we will encode both chords and some representation of their sequence as features for our model.

Because there are hundreds of potential genres, we limit our space to the following defined genres - 'Classical', 'Country', 'Disco', 'Jazz', 'Modern' (inclusive of Pop and Rock genres) - so that we can work with a more defined data set. This small subset of genres allows us to limit the scope of our research, as well as comment on our aforementioned hypothesis based on conclusive experimental evidence. These genres were selected to provide examples of both very different genres (classical and pop) and potentially similar genres (disco and pop).

4.2 Database

The database we used is The Lakh MIDI Dataset (Raffel, 2016a) (Raffel, 2016b). This database has some pre-labeled examples for genre, which will be very useful for our project. It is also large enough that we can split off a specific percentage of our data-set as training data, while the rest is divided into cross-verification and test data. The MIDI data of these files will be used to extract the actual feature data and pair them with labels.

However, the data-set genre labels are user generated, so there is not a firm consensus as to their accuracy. As we were reviewing the data set for troubleshooting, we did find several examples in genres that did not even remotely seem relevant, but the data set was too large to sort through and resolve all of these individually. This introduction of noise almost certainly detracted from our accuracy, though the extent of its impact is difficult to determine. Furthermore, there were many corrupted files in the data set. Rock in particular was already somewhat small, so when the majority of the rock files could not produce viable features we combined pop with rock due to their often similar nature.

4.3 Feature Extraction to Usable Information

We ultimately used four main types of features: tempo, instruments, notes, and rhythm.

Tempo in beats per minute was read directly from the XML file, and then placed into one of 7 bins, listed in table 1.

< 40	41-70	71-100	101-130	131-160	161-180	180 <

Table 1: Tempo Categories

Instruments were also placed into 7 categories, which can be seen with some examples in table 2.A full list available in table 6 in the appendix. An unknown category was included to make the system more robust and allow for unique instruments

Brass	Trumpet, Tuba
Strings	Violin, Guitar
Woodwind	Flue, Clarinet
Keyboard	Piano, Synthesizer
Percussion	Drums, Tambourine
Voice	Soprano, Alto
Unknown	Theremin

Table 2: Instrument Categories

Notes and rhythm for a single measure were paired in a 14x24 matrix. The 14 columns represented the 12 scale degrees, plus one for rest and one for un-pitched percussive hits. The pitch {C,D,E,...} of the note corresponds to a different number (scale degree in {1,2,3,...}) depending on the key in which the note is played. For example, in C major, C corresponds to the first spot in the array, C#/Db to the second, and so on, while in G major, C corresponds to the fifth spot. This made it possible to compare chords across key signatures without explicitly labelling them, since a C one chord (1,3,5) would have the same scale degrees as a G one chord (1,3,5), despite being comprised of different notes. Although this ignores chord inversions (3,5,1) (5,1,3), for our purposes it is a firm enough foundation to provide distinguishing characteristics between genres. Archetypal jazz seventh chords, for example, do not rely on inversion to be distinguished from the typical one, four, and five chords of pop.

Like the scale degree and the key signature of the song, the rhythm is adaptive based on the time signature of the song. We only included 4 very common time signatures, listed in table 4. So, for a 4/4 song in C, a quarter note C would turn on ones in first column, for the first 6 rows. An example measure and table is included as follows3. The hit column is excluded for space 3. Notes smaller than 32nd notes were ignored, due to their relative infrequency in real sheet music. In 4/4 time, for example, a quarter note is 1/4 of the measure, so 6 of the 24 slots are flipped to 1. In 3/4 time however, it is 1/3 of the measure, so 8 of the 24 slots are flipped to one. This proved problematic when actually using the test data set, but this will be discussed in more detail in section 5.1.

Using the measure depicted in figure 1 as an example, we can fill out one measure in table 3. As you can see, the whole note in the bass clef causes C, E, and G to be turned on the whole measure, while D and rest are only turned on while they are present. In this way, we encoded the notes, rhythms, and their sequence in one set of features.

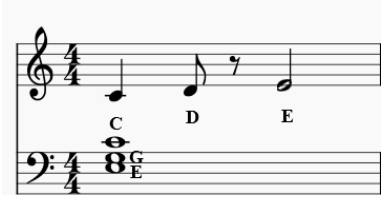

Figure 1: Example Measure (C Major)

C B#	C# Db	D	D# Eb	E Fb	E# F	F# Gb	G	G# Ab	A	A# Bb	B Cb	rest
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	1	0	1	0	0	1	0	0	0	0	0
1	0	1	0	1	0	0	1	0	0	0	0	0
1	0	1	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0

Table 3: Example measure note/rhythm array

Time Signature> Note Type V	4 4	3 4	6 8	2 4
32nd	0	1	1	1
16th	1	2	2	2
8th	3	4	4	6
dotted 8th	4	6	6	8
quarter	6	8	8	12
dotted quarter	9	12	12	18
half	12	16	16	24
dotted half	18	24	24	NA
whole	24	NA	NA	NA

Table 4: Time signature based rhythm allocation

Songs are comprised of many measures, so we chose to feed in sets of 10 or 3 measures at a time along with the combined tempo/instrument information (14x1), resulting in a 14x241 or 14x73 array for each feature passed in to the network. This meant we were able to get many features out of longer songs, and make use of song snippets as short as 3 measures.

4.4 The Chosen Pipeline

Following our analysis from section 3.2, we decided to utilize RNNs due to their temporal capabilities. We used the LSTM modules from the Pytorch library to generate a network where the hidden states and cell weights are initialized with random values. Each LSTM cell has an input, forget, cell, and output gate with their respective weight vectors and activation functions. The operation of each LSTM cell can be represented with the following mathematical equations (PyT),

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\
g_t &= tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\
c_t &= f_t * c_{(t-1)} + i_t * g_t \\
h_t &= o_t * tanh(c_t)
\end{aligned}
$$

where h_t is the hidden state at time t, c_t is the cell state at time t, x_t is the input at time t, $h_{(t-1)}$ is the hidden state of the layer at time $t-1$ or the initial hidden state at time 0, and i_t, f_t, g_t, o_t are the input, forget, cell, and output gates, respectively. σ is the sigmoid function and $*$ is the hadamard product. $(W_{ii}, b_{ii}), (W_{if}, b_{if}), (W_{hi}, b_{hi}), (W_{hf}, b_{hf})$, $(W_{ig}, b_{ig}), (W_{hg}, b_{hg}), (W_{io}, b_{io})$, and (W_{ho}, b_{ho}) are the weights and biases associated with transfer of information between individual gates of the LSTM cell.

For the task of music genre classification, our network architecture consists of LSTM layers, followed by a fully connected and a SoftMax layer at the end to generate prediction probabilities for each genre. A sample LSTM network is shown in Figure 2.

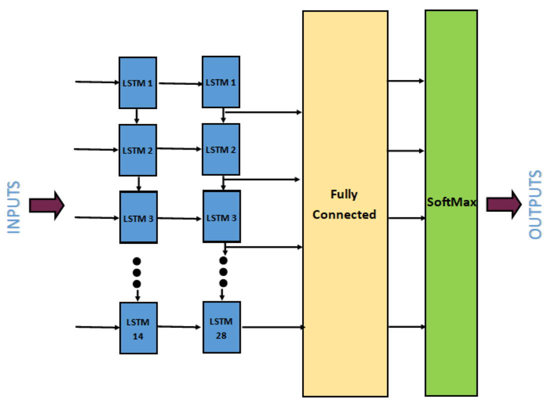

Figure 2: Sample LSTM Classifier Network

We experimented with different numbers and sizes of the LSTM hidden layers to find an architecture which performs with the best possible accuracy. These results are presented and discussed in section 5.1.

Number of measures in data	Two LSTM layers, width 14	Two LSTM layers, width 28	Single LSTM layer, width 28
3	train: 62.84%, val: 63.46%, test: 63.47%	train: 63.08%, val: 64.16%, test: 63.85%	train: 62.98%, val: 63.50%, test: 63.41%
10	train: 63.40%, val: 63.34%, test: 63.13%	train: 63.06%, val: 63.48%, test: 64.88%	train: 63.46%, val: 63.05%, test: 64.23%

Table 5: Classifier Accuracies

5 Discussion

5.1 Results and Analysis

We tried varying the structure of the RNN in both number of hidden layers and number of hidden dimensions, but the accuracy did not vary by more than a few percent. We also tested 10 measure long features and 3 measure long features, and had similar results for both. These values are presented in Figure 5. This could indicate several things. Perhaps the sequential nature was not as important as we thought, or only close range sequence matters, or the RNN structure accounted for the sequence regardless of the number of measures passed in simultaneously. Features such as instrument and tempo that remain constant throughout the song may also be weighted more by the classifier in making its decision.

We also studied the average test error of the classifier for the different music genres we considered, shown in Figure 3. These errors were calculated using samples from the test data in batches of size 1000. From the bar graph, we see that the classical genre had lowest error. This is because it is probably the most distinct from the other genres we used. We also found that the data set we used had a higher proportion of classical songs than of any other songs. Due to a combination of these factors, the classifier learned to distinguish this genre more accurately than the rest. The higher test errors for songs from the other four genres can also be justified since it is possible that they have common features, or mixed music styles that the classifier failed to recognize. To illustrate with an example, 'Uptown Funk' is a song that one may classify as both belonging to the 'disco' and 'pop' genres.

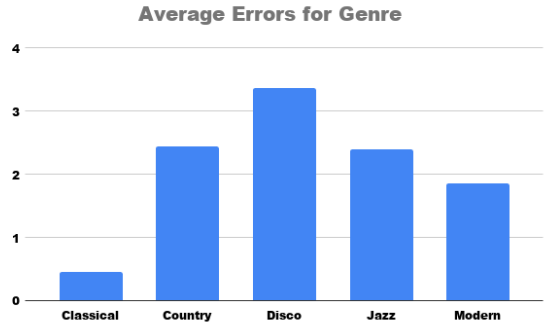

Figure 3: Testing Error per Genre

6 Conclusion

Our overall accuracy of around 63% is higher than the 40% we anticipated from research (Z. Fu and Zhang, 1981) with a similar number of features, indicating the potential of the features we selected. As expected, the network performed the best on the most distinct genre (classical), and had the most difficulty with disco, a genre from which many modern songs borrow elements.

6.1 Future Work

Possible future work includes comparing both the SVM and RNN results instead of just picking one. Determining which of the features do the best job individually to distinguish the different genres and thus, which we can ignore when building future networks. Additionally, for our data conversion, time signature changes beat length for storage. This works, but is not optimal for feature comparison, as it makes note patterns which should look rhythmically the same look different in different time signatures.

In other feature based expansion, our handling of rests has significant potential to be improved. When handling rests, we currently offset the rest based on x-locations of nearby notes such that the ordering of the rests compared to the notes is correct. However, many of the MIDI files from the data set we used were machine-converted from audio, meaning they're not 100% accurate to the original sheet music. This leads to issues such as doubled up rests, 128th notes, and strange offsets. To account for this we ignored any notes which were offset past the end of the measure. Obviously this is not ideal, and finding some way to utilize this data would potentially improve the training data for rhythm by a large margin.

Additionally, while we handled the features we thought had the most potential as our inputs, there are many more features which could be explored in this mid-level. One example would be to include more specific instruments, instead of the broad categories we chose, since some instruments are highly specific to certain genres. Some other mid level features, such as dynamics, would be difficult to encode from XML data, but could provide a nice augmentation to research done in audio based fields.

Another option would be to compare based on time-period in one genre, seeing if, for example, modern jazz is drastically different from older jazz. Release date of song could also be passed in as a feature, and could be very helpful for genres such as classical and disco.

Final Thoughts

6.2 Project Video

Link to our project video.

Acknowledgments

We would like to thank Dr. Roth, the teaching assistants, MuseScore (Mus) and this institution for assistance with this project.

References

Musescore. https://musescore.org/en. Accessed: 2019-10-25.

Pytorch. https://pytorch.org/docs/stable/nn. Accessed: 2019-11-20.

H. Bahuleyan. 2018. Music genre classification using machine learning techniques.

D. Rizo C. Pérez-Sancho and J. M. Iñesta. 2009. Genre classification using chords and stochastic language models. *Connection Science*, 21(2-3):145–159.

W A. De Noni M. G. Armentano and Hernán F. Cardoso. 2016. Genre classification of symbolic pieces of music. *Journal of Intelligent Information Systems*, 48(3):579–599.

D. P. W. Ellis M. I. Mandel. 2005. Music genre classification using machine learning techniques.

C. McKay. 2004. Automatic genre classification of midi recordings, master of arts thesis.

C. Raffel. 2016a. The lakh midi dataset v0.1. https://colinraffel.com/projects/lmd/. Accessed: 2019-10-25.

C. Raffel. 2016b. Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching.

K. M. Ting Z. Fu, G. Lu and D. Zhang. 1981. A survey of audio-based music classification and annotation. *IEEE Transactions on Multimedia*, 13(2):303–319.

Appendix

Full instrument categorization

Brass	trumpet, trombone, horn, euphonium, tuba, cornet, flugelhorn, sousaphone,mellophone, bugle
Strings	guitar, bass, violin, viola, cello, banjo, mandolin, ukulele, harp
Woodwind	piccolo, flute, oboe, Cor anglais, clarinet, saxophone, bassoon, contrabassoon, contrabass, bagpipes, recorder, vuvuzela, shenai, shehnai, harmonica, bandoneon
Keyboard	piano, synthesizer, harpsichord
Percussion	timpani, drumset, drums, cymbals, triangle, tambourine xylophone, marimba, vibraphone, bells
Voice	soprano, alto, tenor, bass
Unknown	left blank, added as necessary

Table 6: Instrument Categories (Complete)