Abelardo Riojas
Scientific Computing Practicum Spring 2022
Prof. Anke Meyer-Baese

Report Draft

Introduction/Background:

Music Information Retrieval is the interdisciplinary science of extracting musical information from audio files. MIR draws from an array of academic fields such as: musicology, digital signal processing, computational intelligence, psychoacoustics, and machine learning. In the past MIR was accomplished using classical machine learning techniques such as decision trees, random forests, regression, and clustering using features extracted via DSP techniques. Features used in classical MIR range from low level features (that make sense mainly to the computer) like zero crossing rate, amplitude envelope, and root mean square energy to high level features (more recognizable to humans) like tempo, key, and time signature.

With the advent of image classification models using neural networks, we can use models like CNNs to extract features automatically and perform inference. Provided we have a way to convert audio files into images, we can apply the same techniques used in image classification to music. One way of converting audio to images is to apply a spectrogram to the audio file. The most popular type of spectrogram used for MIR analysis is the mel spectrogram, along with its brother the MFCC (mel frequency cepstral coefficient). The mel scale, (after the word melody) is a perceptual scale of pitches judged by listeners to be equal in distance from one another. Converting the frequency axis to mels allows the spectrogram to approximate the human auditory response much better than the linear spaced frequency bins used in normal spectrograms.

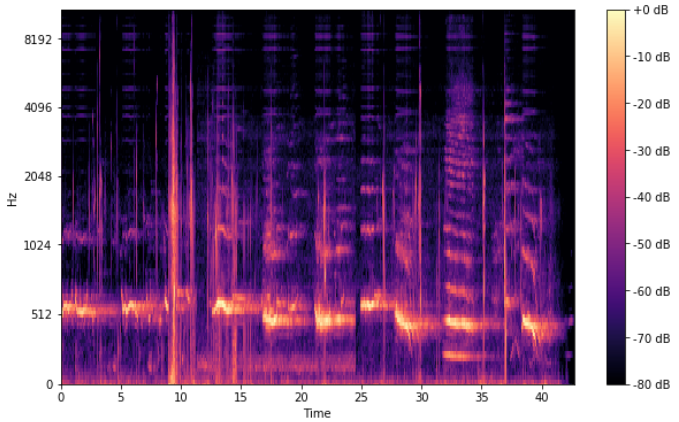


Figure 1: Example of a mel spectrogram.

Abstract:

The goal of this project is to use Convolutional Neural Networks to classify songs into musical genres. We compare the use of mel spectrograms and MFCCs on the accuracy of the model. We also investigate the effect of adding first and second derivatives on the spectrograms as channels. In addition, we also investigate the length (in seconds) of the audio samples' effect on the accuracy.

The classification this model performs is different from most music genre classification models. Instead of predicting one and only one genre, the model allows for multiple genres to be chosen as a final classification result. Since musical genres are based on creative subjectivity, this allows for more robust classification of songs. In the same way a movie's genre can be ['Horror'], ['Comedy', 'Romance'] or ['Action', 'Thriller'] our CNN model can classify songs into genres such as ['Electronic'], ['Psychedelia', 'Rock'], or ['Hip Hop', 'Country'].

Previous research has shown that CNNs are the best model to use for music genre classification above the traditional ML methods. The rationale for this research extends to the possibilities of using these classifications for music recommender systems such as Spotify's Discover Weekly.

Methodology

Gathering the Data

In order to perform genre classification, first we need a bunch of music files. There is the already existing GTZAN dataset, which contains 100 songs for 10 genres (Blues, Classical, Country, Disco, Hip Hop, Jazz, Metal, Pop, Reggae, Rock). With each song only having one possible label, and the range of genres not as wide as they could possibly be, I decided to gather my own music files in an attempt to make a better version of this dataset.

The genres themselves are based off the list of main musical genres from RateYourMusic.com. I filtered out the genres Blues, Comedy, Field Recordings, Musical Theater and Entertainment, Industrial Music, New Age, Metal, Regional Music, Ska, Spoken Word, and Sound and Effects as they either:

A: Did not represent a genre of music popular enough to be a helpful classification. (Sound and Effects, New-Age, etc.)

B: Is a genre that is contained by another genre. (Metal, Blues)

C: Genre contains too broad of a musical range (Regional Music)

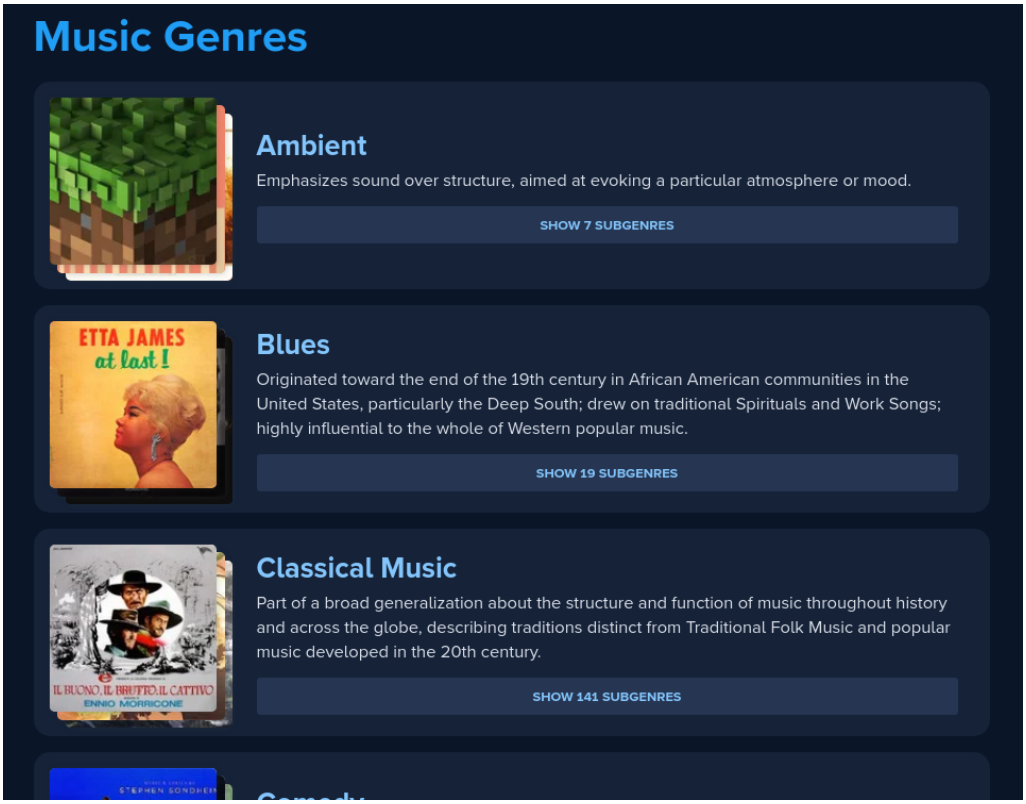


Figure 2: RYM genres

This leaves us with a final list of genres as follows: Ambient, Classical, Country, Dance, Electronic, Experimental, Folk, Hip Hop, Jazz, Psychedelia, Pop, Punk, RNB, Rock. As well as any two-pair combination between these genres deemed valid by representation in music playlists online.

Valid two-genre fusions are as follows.:

Ambient-Electronic, Ambient-Experimental, Country-Rock, Country-Hip Hop, Dance-Electronic, Electronic-Experimental, Electronic-Hip Hop, Folk-Rock, Jazz-Hip Hop, Psychedelic-Rock, RNB-Hip Hop, and the combination of Pop and the genres Country, Dance, Electronic, Folk, Hip Hop, Punk, RNB, and Rock.

The songs were gathered via searches on the Spotify app like ‘Ambient playlist’ or ‘Pop Rock 2000s.’ To make sure the full range of the genre was expressed in the dataset, I consulted RYM again for details on the list of sub-genres for each main genre, and found songs representing each equally. For example: songs under the ‘Rock’ genre contain music from the Classic Rock of the 60’s and 70’s, Progressive Rock and Metal from the 80’s, the Alternative Rock of the 90’s and 00’s, as well as newer Indie Rock from the 10’s and 20’s.

The songs were also listened to and filtered by myself to exclude non-examples that may have slipped in.

Once the full dataset playlist was made, songs were downloaded off of the internet using the DeezerAPI and the scriptable music downloader StreamRip. Playlists would yield about 70% of the songs after downloading. Karaoke, cover, or piano/violin renditions of songs were filtered out of the dataset as well. The file format of the audio is .FLAC with a sampling rate of 44.1 KHz (twice that of the GTZAN dataset). With higher quality music files in the correct genre format that we want, we are ready to start preparing the data.

Preparing the data

Each .FLAC file in the dataset is converted to a 30 second audio sample. With 10 seconds each coming from the quarter, half, and three-quarter section of the file. From there, we choose the length used for the training examples.

Sample length	Number of training/testing examples	Shape of example: Mel Spectrogram	Shape of example: MFCC
30 seconds	3200 samples	(128, 2584, 1)	(13, 2584, 1)
10 seconds	9600 samples (x3)	(128, 862, 1)	(13, 862, 1)
3 seconds	320000 samples (x10)	(128, 288, 1)	(13, 288, 1)

There are also versions of the dataset with first and second derivatives for each training example.

Sample length	Number of training/testing examples	Shape of example: Mel Spectrogram (With derivatives)	Shape of example: MFCC (With derivatives)
30 seconds	3200 samples	(128, 2584, 3)	(13, 2584, 3)
10 seconds	9600 samples (x3)	(128, 862, 3)	(13, 862, 3)
3 seconds	320000 samples (x10)	(128, 288, 3)	(13, 288, 3)

Along with derivatives, we experiment with the use of audio augmentation. Pitch shifted and noisy versions of songs were also generated and made spectrograms of.

With all the different parameters (augmentation, deltas, type of spectrogram, and length of sample), there exist 24 different versions of this dataset.

Total number of training examples	Augmentation	Deltas	Type of spectrogram	Length of sample (in seconds)
3200	False	False	Melspec	30
9600	True	False	Melspec	30
3200	False	False	MFCC	30
9600	True	False	MFCC	30
3200	False	True	Melspec	30
9600	True	True	Melspec	30
3200	False	True	MFCC	30
9600	True	True	MFCC	30
9600	False	False	Melspec	10
28800	True	False	Melspec	10
9600	False	False	MFCC	10
28800	True	False	MFCC	10
9600	False	True	Melspec	10
28800	True	True	Melspec	10
9600	False	True	MFCC	10
28800	True	True	MFCC	10
28800	False	False	Melspec	3
86400	True	False	Melspec	3
28800	False	False	MFCC	3

86400	True	False	MFCC	3
28800	False	True	Melspec	3
86400	True	True	Melspec	3
28800	False	True	MFCC	3
86400	True	True	MFCC	3

Attempts to process these datasets using JSON files (to preserve the resolution of float arrays for the spectrograms) failed. System memory would often kill the process, even on the Bones, the system located in the Department of Scientific Computing which has 64 GB of RAM. As a work around, the spectrograms were processed as .PNG images (which turned the array values into integers) and put in a folder.

Tensorflow ImageDataGenerators were used to load the data into the machine without crashing the PC.

```
df = pd.read_csv(f'images_{int(aug)}_{int(deltas)}_{int(mfcc)}_{slices}.csv')
df = df.sample(frac=1)
columns = 'Ambient Classical Country Dance Electronic Experimental Folk HipHop Jazz Pop Psychedelia Punk RNB Rock'.spl

datagen=ImageDataGenerator(rescale=1/255.)
test_datagen=ImageDataGenerator(rescale=1/255.)

n_rows = len(df)
cm = "grayscale"
if deltas:
    cm = "rgb"

train_generator=datagen.flow_from_dataframe(
dataframe=df[:int(n_rows*.8)],
x_col="Filename",
y_col=columns,
batch_size=16,
color_mode = cm,
seed=42,
shuffle=True,
class_mode="raw",
target_size=(shape[0], shape[1]))
```

Figure 3: Tensorflow image data generator.

Building the model

There are two different (albeit very similar) CNN model architectures used in this study. One for MFCC spectrograms and the other for mel spectrograms. Because MFCCs have such a low height, the kernel shapes and pooling sizes had to be adjusted in the MFCC architecture to ensure the tensors could flow through the model correctly.

To arrive at the final model architecture, many different architectures were experimented on using the no augmentation, no deltas, mel-spec, 3 second sample dataset.

This architecture (below) converged to a final validation accuracy of 65% percent, and was the model I used for the study.

```
    model.add(Conv2D(8, kernel_size=(3,3), strides=(1,1), input_shape = shape))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(16, kernel_size=(3,3), strides=(1,1)))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(32, kernel_size=(3,3), strides=(1,1)))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(64, kernel_size=(3,3), strides=(1,1)))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(128, kernel_size=(3,3), strides=(1,1)))
    model.add(BatchNormalization())
    model.add(Activation('relu'))

    model.add(Flatten())
    model.add(Dropout(rate=.3))

    model.add(Dense(14, activation = 'sigmoid'))


model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.summary()
```

Figure 4: Model architecture (Mel spec)

The final dense layer of 14 outputs uses the sigmoid activation function instead of softmax because we are doing multilabel classification. Each genre is treated as an individual bernoulli process, and the criterion for a label classification is a value over .5.

We use binary cross entropy as the loss because it compares each of the predicted probabilities to actual class output which can be either 0 or 1.

The optimizer used was the standard Adam optimizer, as it is a complex, reliable optimizer that converged well (without any modifications) on the tested dataset.

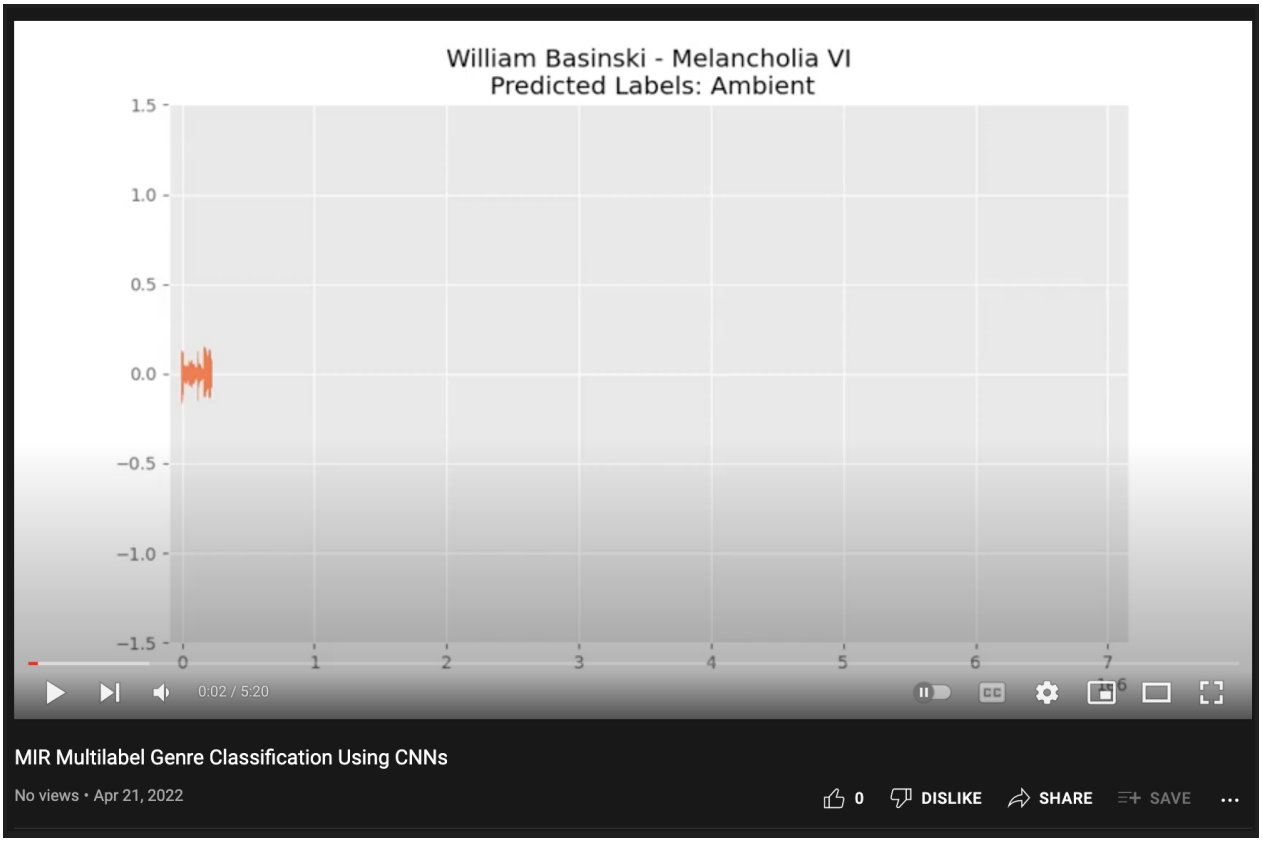
Results

Maximum validation accuracy	Augmentation	Deltas	Type of spectrogram	Length of Segment (in seconds)
36.88%	No	No	Melspec	30
39.79%	No	No	Melspec	10
65.13%	No	No	Melspec	3
70.62%	Yes	No	Melspec	30
69.34%	Yes	No	Melspec	10
64.54%	Yes	No	Melspec	3
71.46%	Yes	Yes	Melspec	30
67.07%	Yes	Yes	Melspec	10
64.56%	Yes	Yes	Melspec	3
70.72%	Yes	Yes	MFCC	30
61.80%	Yes	Yes	MFCC	10
50.90%	Yes	Yes	MFCC	3
66.45%	Yes	No	MFCC	30
59.89%	Yes	No	MFCC	10
50.05%	Yes	No	MFCC	3
31.90%	No	No	MFCC	30
34.48%	No	No	MFCC	10
31.90%	No	No	MFCC	3
36.56%	No	Yes	Melspec	30
38.95%	No	Yes	Melspec	10

45.59%	No	Yes	Melspec	3
29.06%	No	Yes	MFCC	30
33.33%	No	Yes	MFCC	10
40.48%	No	Yes	MFCC	3

Results:

Below is a YouTube video of example classifications on songs outside of the training and validation sets. As you will see, it does its job quite well!



Click me

Conclusion

To conclude, I would like to speak on just how much I learned doing this project. The majority of the work was done by myself, with the help of my

mentor Nathan Crock and numerous articles on machine learning posted online by various contributors on the websites TowardsDataScience.com.

Making a dataset is hard. Making a data pipeline is hard. Machine learning is notoriously hard. So many obstacles had to be overcome to deliver a final product I was proud of.

If I had to restart from scratch, I would make a version of this AI with a data driven approach, perhaps using Gaussian Mixture models to cluster the spectrograms into groups and classify new spectrograms into the groups.

This is my last assignment for the Department of Scientific Computing here at FSU. I am proud of my work here. Especially this work, which shows how driven I am to finding solutions to creative problems.

Thank you, DSC.

- Abelardo Riojas