Music Generation - Functional Spec

Background

The problem we're trying to solve. Why it's important. How users will benefit.

Fundamentally, we want to leverage machine learning (ML) to provide a way for users to **generate custom music for their specific use cases**. See the section below, *Users and Use Cases*, for examples of these.

Users, of course, have options for obtaining music besides using ML to generate their own. However, upon further exploration, these options aren't as workable as they might at first appear.

- Using **copyrighted music** without permission is both unethical and illegal. Getting permission to use copyrighted music can be onerous and expensive, and in fact, might dead end with the copyright holder simply refusing to grant permission.

- Negotiating with **composers and musicians** to create custom music has the potential to be a costly--and perhaps more importantly--time-consuming process. Also, there is no guarantee that once the music is created it won't be encumbered by licenses that make it difficult to use beyond the initially specified use case.

- With the advent of **open licenses**, such as Creative Commons, a lot of unencumbered music is now available on the Internet. However, the main problem here is *discovery*: How does the user find music that is a good fit for their use case. Personal experience indicates that this is surprisingly difficult.

In contrast, what we offer is an experience where the **user provides a sample** of music, which is similar to what they seek. Our package uses this sample to generate a stream of music--of

arbitrary length--that is **similar to the sample but free of licensing restrictions**. The user can then edit this music stream with standard audio editing software[1] to customize it for their use case.

[1] Audacity, Adobe Audition, Apple GarageBand

Users and Use Cases

Who will use your system? What level of computer experience do they require? What domain knowledge must they have?

The users of our package will be anyone who is interested in generating copyright-free music but doesn't have the necessary knowledge in music theory or the technical skills required to build/train ML models to compose music. The following are some examples of our target users:

Podcast Producers
Podcast producers include musical elements in their podcasts to make them more engaging to their listeners. Using intro/outro and background music to move the story forward contributes to a professional sounding podcast. But obtaining licenses to music can be expensive and might not sound original, so producers can use our package to generate their own unique music. They will need to have basic Python knowledge to install and use our package, as well as how MIDI/MusicXML files work.

Video Creators
Music is an extremely important element in video production. Video creators can generate music for their workout, cooking or traveling videos using our package and edit it to match the pace or mood of their videos. These users need to know the basics of Python to install and use our package. They'll also have a basic understanding of how MIDI/MusicXML files.

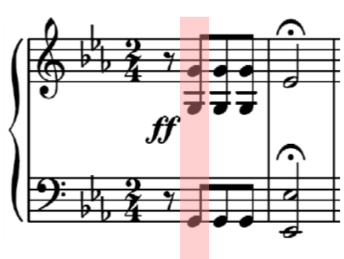
Music Enthusiasts
Music enthusiasts who dream of composing their own music but do not know how to use instruments can generate music similar to their taste using our package. The computer experience they should have is basic Python to install and use our package, as well as an understanding on how MIDI/MusicXML files work.

Data

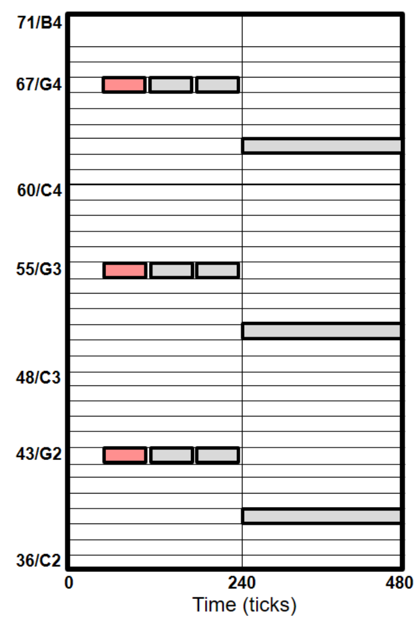
What data you will use and how it is structured.

The primary dataset that we will be using for our project comes from the Lakh MIDI dataset curated by Colin Raffel(See Colin Raffel. "Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching". PhD Thesis, 2016.) The model that will be shipped in the final product will be trained on a subset of the Lakh MIDI dataset. The reason why only a subset of the data was chosen was due to the limited computational resources available to us to train the model. The data consists of 176,581 MIDI files of varying genres. The dataset itself is derived from the Million Song Dataset(see Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.). That is, the songs in the Lakh MIDI dataset align with a subset of the Million Song Dataset. The structure of some example MIDI data is presented below. MIDI notes are separated into note on and note off events with a corresponding time that the notes occur. In addition, additional data can be parsed from MIDI messages such as the velocity of the note(note volume).

Figure 1.13 from [Müller, FMP, Springer 2015]



Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	63	100
0	NOTE ON	2	51	100
0	NOTE ON	2	39	100
240	NOTE OFF	1	63	0
0	NOTE OFF	2	51	0
0	NOTE OFF	2	39	0



User Experience (UX)

How users will interact with the system and how the system will respond.

The **easy_music_gen** package exposes the following constructor and method.

```
from easy_music_generator import easy_music_generator as emg

music_generator = emg.EMG( )

music_generator.analyze( sample_dir )
```

sample_dir: str

A list of MIDI and Music XML files to use as samples. The list can include any number of MIDI files, MusicXML files, or combination of these. The filetype is inferred from the extension: MID or MIDI for MIDI files, MXL for MusicXML files..

```
music_generator.generate( bars, output_dir )
```

bars: int

The length of the outputted music in number of bars.

output_dir: str

Output directory for the generated MIDI file.