A Layperson's Introduction to Computational Music Analysis and Generation

Alexander Lill

Abstract—From the advent of computers and digital representations of sound, there has been interest in how computers may be used to process, interpret, and create new music. Machine learning (among other fields) has significant impacts on how composers (and researchers) can analyze and generate music, even including specific libraries and technologies to ease the process.

Index Terms—Artificial Intelligence, Machine Learning, Music, Composition, Analysis.

1 Introduction

Western music has a long history of analysis, research, and intellectualization that aims to catalogue and define much of the nuance involved within the artistic practices of composition and performance. Composers, for example, may ask how certain orderings of chords produce specific emotional effects, or performers may have an interest in how the subtle details of style can be studied to apply within their own performances. Naturally, with the advent of computers as a tool in a researcher's arsenal, the question arose: How can one process sheet music, audio, etc. in a way that helps analysis and answers analytical questions posed above? Or similarly, how can one utilize computational power as a method for automatically generating new compositions based on a specific data input?

In this paper, I aim to motivate the reasoning behind studying the computational analysis and generation of music, introduce and teach on MIT's *music21* as an example of a popular and useful library for computational analysis, and give a broad swath of summaries of papers that famously or interestingly utilize computational processing of music as their main focus to inspire Western music-oriented programmers to begin their own explorations.

1.1 Challenges

Some parts of musical analysis lend particularly well to computational assistance, such as asking the question *"How many times does a particular sequence of notes occur?"* Other questions, however, such as *"How can we quantify a 'good' composition?"* are far more difficult to answer and generate a heuristic for. Western music is also filled with all kinds of ambiguities, where two professionals may have different interpretations of what the analysis a particular chord or melody represents, where neither of them is wrong. Navigating these challenges is a fundamental part of computational music. Furthermore, for large-scale projects, there is often difficulty in finding an appropriate dataset unless one is dealing with a well-researched problem, and if in a specific field, many researchers are left to create and label a dataset on their own. Or, if a dataset is already available, there is often a barrier in the form of some significant financial burden to acquire copyrighted data or to find a free alternative for public use. With an introduction to music21, I hope you may be motivated to pursue individual projects or use their convenient dataset, and if not, one can see how these challenges were navigated in a series of papers from a variety of musical fields.

2 *music21*: A PYTHON LIBRARY FOR MUSICAL ANALYSIS AND ENCODING

To cite their own website, "Music21 is a set of tools for helping scholars and other active listeners answer questions about music quickly and simply" [8]. It provides tools for programmatically dealing with Western music, whether for writing, analysis, or any sort of computational analysis, whether for making images of sheet music to display or to find instances of tone rows in 12-tone music.

music21 also provides methods to easily load an astounding number of music from the corpus of a variety of composers, such as Bach, Handel, Beethoven, Chopin, Joplin, and more from the Classical canon, allowing fans of Western Art music to immediately dig into pieces that they undoubtedly have heard before.

2.1 Installation & Introduction

While the official music21 document provides a more in-depth beginner's guide, I aim to get interested readers quickly running and distill some of the major data types into simpler terms to get them up and running, leaving more details and nuance to their official channel. music21 requires Python 3.7+, and is installed via pip: `pip3 install –upgrade music21` After this, it can be imported.

2.2 The Note Class

The base item within music21 is the `note` class, which has its pitch specified using standard keyboard notation. For example, `tmp = note.Note(''C4'')` stores middle C into the variable `tmp`. You can also get the pitch, the pitch class, the frequency, the octave, the name, and more from an instantiated note class, which may be particularly convenient depending on the type of work you're doing. Watch out: One quirk with the instantiation is that while sharps are notated as you'd expect ("`C#4`"), flats are written using a "`-`": "`C-4`". Finally, rests are created by calling the "`note.rest()`" method.

2.3 The Duration Class

A standalone duration object can be instantiated with either the American nomenclature of "`whole`", "`half`", ..., "`8th`", etc. OR with a double representing the amount of quarter notes they occupy. For example, `duration.Duration(''half'') == duration.Duration(2)`, and one can instantiate varying lengths like a dotted sixteenth with an argument of `.75`.

The `note` object from above has both a pitch and a duration class as an attribute, meaning that once you instantiate a note, you can set its duration of a given note object by changing its `.duration` member.

2.4 Organizing Notes

For simple use, one is fully available to leave their collections of notes represented with the standard Python list type. music21, however, also provides the `Stream` object with the subclasses of `Score`, `Part`, and `Measure`. `Stream` objects can store notes (or any other subclass) with an offset from the beginning, meaning that a `Stream` containing a whole note and a quarter note will note them as being at offsets 0.0 and 4.0 respectively.

One can imagine the hierarchy of a given piece of music being a `Stream` object, which contains a `Score` object, which contains `Part` objects, which each have `Measure` objects, which each have a `Stream` of `Note` objects. This is a given example for how something like an imported Bach chorale would appear.

Some simple operations include instantiation with `stream.Stream()`, appending notes via `.append(note)`, and checking the legnth of a stream via python's standard `len(stream)`. A text output (including the quarter note offsets) can be generated with `.show('text')`, as well as other representations via other arguments.

2.5 Chords

Chords objects can be instantiated similarly to notes, but with a list of arguments. For example, an F major

triad starting on F4 may be written as `f = chord.Chord(['F4', 'A4', C4])`. There are a variety of methods to access information about chords that I will not list here, but for example, one can access the root, third, fifth, seventh, etc., add or remove individual pitches, get a common name for the chord, and more.

2.6 Accessing Corpora

music21 provides easy access to a large library of works in the musicXML format, allowing for an interested reader to immediately begin analysis or computation involving the works of Bach, Haydn, and more. One can search for available items of Bach, for example, with the `corpus.getComposer('bach')` method. Once one has decided on a filepath, they can load a score with the `.parse([path])` method, with either a selection from the default provided scores or an appropriate musicXML file.

2.7 music21 Conclusion

Although only providing a surface level introduction, the above should be plenty to start analyzing, writing, or generating music via Python code. music21 has an *extremely* rich feature set, including a dizzying amount of tools specific for different kinds of Western music analysis, whether tonal, atonal, 12-tone, etc. With the understanding of notes, durations, and streams, there is plenty of room for one to start exploring their own projects that, on one hand, aren't dealing with frustrating homebrew implementations of representing music, but also without spending too much time digging through the music21 tutorial which was written for a more casual audience programming-wise. I hope it may help you jump in after you become inspired by some of the projects which are summarized in the coming pages.

3 Applications in Jazz

Jazz is a broad spectrum of American indigenous music that often features improvised performance. These performances have a history of nearly one hundred years that shows the growth of incredible improvisatory performers, with stylistic shifts in technique and harmony. Naturally, as a kind of chaotic language within the musical world, attention has been drawn to computationally analysis on the improvised material to better understand it and to generate interesting music algorithmically. But as mentioned before, defining what "interesting" means has some challenges, as improvisation has a broad variety of approaches that are all artistically valid. Two highly differing approaches to homphonic jazz solo generation follow below.

3.1 Machine Learning of Jazz Grammars

Gillick et al.'s *Machine Learning of Jazz Grammars* demonstrates how a model that automatically detects structures (called *slopes*) alongside a pretraining process can yield generative probabilistic grammars that are representative of a specific class of input (such as solos from a specific player) [5].

The authors note that as a jazz solo provides a mixture of regular and surprising lines, one can "generate a melody by first generating an abstraction of the melody, and then *instantating* that abstraction" into an actual playable melody. In short, lines within an input solo are broken into chunks, notes and phrases are categorized within certain classes, a clustering algorithm is applied to the output data, then a Markov chain is trained on the probability ech cluster is moved between, and a final model is output.

Gillick et al. went the route of using qualitative analysis to define the quality of the model, and in blind tests, listeners could fairly accurately match grammar-generated solo phrases to the players they were meant to imitate. However, the authors note that while smaller phrases are coherent, they lack the large-scale structure and phrase development of actual professional solos.

Where many modern models tend to focus on neural net implementations, I find it particularly interesting to see a model which avoids neural nets and automatically genereates grammars. Even if one isn't attempting to generate

homophonic jazz solos, it may be fruitful and interesting to either parse grammars from other genres of music or to build them by hand as an interesting compositional experience.

3.2 BebopNET: Deep Neural Models for Personalized Jazz Improvisations

With BebopNET, Hakimi et al. aim to train a machine learning model that can 1. Generate improvised jazz phrases, and 2. Train that model on user feedback to personalize the improvisations towards individual listener's tastes [11]. They achieve this using a three-layer LSTM network and collection of listener data via a continuous response digital interface (CRDI), which is used in later stages of training to fine-tune the model towards a particular listener's ear. The data set is a collection of improvised solos in $\frac{4}{4}$ meter in the *musicXML* format, which provides chord symbols alongside pitch and rhythm information. Additionally, the data is augmented by transposing the dataset into all twelve keys.

The method for collecting listener preference data involves listeners changing the position of the CRDI dial when "[they] hear a sequence of notes [they] consider to be more (or less) pleasing than those [they] heard previously." With this time-series of information, a beam search is used to find the optimized model candidate, attempting to maximize the characteristics of the liked portions and minimize the characteristics of the disliked portions.

Hakimi et al. conclude by describing the increase in average listener satisfaction when compared to the pre-trained model, showing an interesting (and successful) twist on the commonly-done solo generation paper. As far as limitations, they note that while the solos are fairly coherent, they still lack the macro structure that is apparent in professional human structures, and that while it may be fixed with a sufficiently large corpus, the only viable way to do so would be with the creation of technology to generate transcriptions of raw audio.

While there are many neural net implementations of solo generation (BebopNET itself being an improvement on prior works), their approach of collecting listener data and tuning the model in response is a creative spin on the tech. Although it might be infeasible to test multiple subjects in one's casual personal project, the approach could also be particularly interesting if one trained any kind of model based on how their own individual reactions.

3.3 Performing Structured Improvisations with pre-trained Deep Learning Models

The above papers focus on generating standalone works without any real restrictions on the computational time required to generate them. This isn't without limitation, however, as it inherently makes using those models during a live performance basically impossible. Additionally, while BebopNET aims to tailor itself towards a listener's preferred styles, it (and other implementations) fail to tailor themselves towards a particular performer's individual style [10]. Using a MIDI keyboard and a pre-trained model, Castro can parse real-time input from a performer and generate a drum beat that can improvise with a performer in a call-and-response style, allowing a human player to share a creative improvisation with a robot drummer in real time.

There is still much to be explored with partnering with algorithmic performers, but I think a particularly useful takeaway is the feasibility of generating real-time responses via a MIDI keyboard and quantization of performed notes into a sixteenth-note grid when performed with a metronome. It could be extremely interesting for one to tailor not only drums, but other instruments or harmony in response to performed music.

4 Applications in Western Art (*Classical*) Music

Western Art Music, also colloquially known as classical music within English-speaking Western countries, is a form of music that has a rich and well-defined tradition extending hundreds of years. Although the harmony developed to include new additions and patterns leading into the 20th century, the fundamental

concepts of how harmony is organized easily reaches back to the 1700s. The common movement between tonic and dominant chords and their related paths (as well as how melodies are shaped relating to it) follow a set of fairly consistent rules, making them a relevant choice for computational analysis.

4.1 Google's *Bach Doodle*

In celebration of Bach's 334th birthday, Cheng-Zhi et al. developed a system of input that allow users to input a melody and a have a machine learning algorithm attempt to harmonize it in the Baroque style of J.S. Bach. Their goal was to "extend creative abilities" using models that "rapidly fill in missing parts of [a] composition" [4]. To achieve this, they represented the four-voice counterpoint as a three-dimentional tensor that denotes instruments, time steps, and pitches, but as the model was trained with the goal of representing four-part harmony, it was always given that there are exactly four instruments. Therefore, a melody and its harmonization are represented by a sequence of tensors denoting what note occurs in a specific voice at a specific time.

To train this model, they take a model example from a chorale from Bach's corpus, choose how many varibles (or voices) to erase notes from, and then choose some uniform distribution within each voice to erase [4]. Thus, the model is trained to predict what notes are missing based on the available information, and compare results to the actual realization that Bach chose. After training was completed, the model would take the user-input melody and predict the harmony as in training, but now always with three notes missing.

If one is interested in using the data for their own experiments, it should be noted that a signficant selection of user-generated harmonizations and their respective metadata was made available with the paper.

4.2 Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset

Hawthorne et al. present a new dataset that allow for the direct generation of musical audio based on the creation of new, highly accurate data recorded from an international digital piano competition. They note that music is highly difficult to generate as there is a long-term level of consistency that is difficult to replicate, and while there was much success for shorter periods of time, results would be clearly distinguishable from real audio due to various artifacts [1].

To generate the new dataset, Hawthorne et al. partnered with an piano competition using Yamaha's *Disklavier*, a system for Yamaha factory pianos that allows for the recording and playback with high precision of the velocity of key strikes and pedal positions that a performer uses [3]. This was paired with audio recordings from the real competition which were then cleaned up to provide a time-matched MIDI file and audio recording, creating the largest existing dataset of its kind.

This data was used to train a transcription model that could generate MIDI rolls from given piano audio and create a generation model that could synthesize highly realistic audio from a given MIDI piano roll. The authors note that most piano synthesis is done by the combining and overlapping of sampled piano recordings, which fails to capture much of the intricacy generated by the interactions between different notes from a live piano. The trained model, however, could far more accurately capture them, and interestingly enough, even recreated non-piano sounds such as the breathing of the performer or the sound of movement on a seat. In testing, listeners chose the generated output as their favorite at a rate that closely tailed the actual live recordings.

Not only did Hawthorne et al. make a highly successful model, the effort that went into creating the fine-tuned dataset didn't go to waste, as it is free and available to the public for personal projects or further research. While this was successful for piano, they do admit that it will be a challenge for other instruments, as there aren't such massive and perfect datasets for other instruments (such as winds) that aren't as conveniently able to be quantized in the same way a disklavier performance would.

4.3 Multi-modal Deep Learning for On-line Music Following in Score Sheet Images

In their thesis, Florian Henkel demonstrates a model that can directly track the live performance of a piece of sheet music based on an image of the score for the music. One use of this that they demonstrate is to use an automatic page turning system alongside the model to create a system that can turn a page the instant that a performer gets to the middle of the bottom line of music on a page, allowing them to continue performing without having to manually turn the page or rely on an assistant during the performance. Additionally, the goal was to perform this without some symbolic representation of the score (such as a MIDI file), and instead rely only on images of the sheet music [2].

While other implementations (such as above) provide the means to track the progress of music whenever an intermediate format is used as an aid, Henkel notes that they can often be too expensive, time-consuming, or otherwise inconvenient to obtain, meaning that there is a limited usefulness to their applications outside of the most commonly-performed music.

Henkel's paper (which I have very briefly summarized) demonstrantes an interesting link between the fields of computer vision and of audio analysis, two areas that can be tied to music that aren't directly related to the symbolic processing and generation of musicXML or MIDI files.

5 Other Genres

5.1 Scream Detection in Heavy Metal Music

In their paper, Kalbag and Lerch aim to take recordings of metal music and identify what vocal lines are sung with a clean voice and what lines are screamed, as well as classify what kind of screams are used within various recordings [7].

Part of the challenge came from the fact that there was no existing dataset of metal music with labels identifying the various kinds of screams, between the low, "beast-like" screams and the "high and screechy" [7]. The authors note that this could potentially aid music recommendation systems, as it may help an algorithm more deeply understand the type of screams in metal that a listener favors.

Kalbag and Lerch manually labeled and created a dataset of 281 songs, which was then split into training, testing, and validation data sets. The songs were pre-processed to separate vocals from other instruments and then chopped into short blocks that would be analysed. Using heuristics that are common to Music Information Retrieval tasks, three-label and six-label models were trained to try and identify the types of screams. While the screamed vs clean identification was fairly successful, the specific neural net implementation failed to identify between different types of screams, due to the high levels of similarity between them.

To me, this paper motivates a huge amount of exploration in a fairly approachable package. One could copy their methods of separating vocals, chopping it up, using standard techniques, and training a model to try and pinpoint some kind of heuristic as a fairly approachable personal project. The time involved with labeling would be significant, but as a jumping-off point, Kalbag and Lerch provide a highly digestible approach to audio analysis that could get even a machine learning beginner dipping their toes into an intimidating field. If one has the patience to manually label a dataset (for which the authors describe the method for), they could very realistically attempt to apply the same methods to, say, identify certain musical ornamentations from a jazz vocalist or detect any other effect in a genre that consists of some combination of a vocalist paired with instrument accompinament.

5.2 Melody Harmonization with controllable Harmonic Rhythm

Wu. et al. aims to improve over past harmonization models (such as the aforementioned *Bach Google Doodle*) by training a model that can not only harmonize specific melodies, but also do so with differing harmonic rhythms, the rate at which chords change within the piece [12].

While other models have had a fair amount of success in harmonizing melodies, they often

only consider the time signature 4/4, and cannot harmonize melodies that appear in other simple or complex time signatures. Next, these harmonizations often happen with a fixed harmonic rhythm, an aspect of music that varies between different styles and can often be limiting if held at a constant rate. Finally, Wu. et al aim to allow for user input on how complicated the harmony is, directing it either to be more consonant or more dissonant depending on its input.

Using metrics such as the number of chord types that appear, the entropy of how each type of chord appears, the harmonic distance between a melodic line and its supporting chords, and more [12] and a new encoding method for chords and melodies, Wu. et al not only are able to harmonize uncommon time signatures with new harmonic rhythms, but provide a new level of control over the process. The authors note that while they achieve a higher level of prediction accuracy with their model than other methods, it still lags behind humans, meaning that there is still room for improvement.

I found this paper to be worthy of mention as it generalizes the idea of harmonization away from the rather strictly-held trends that other models often were limited to. Although 4/4 is the most common time signature in Western music (for which data for analysis is readily available), there are many types of Western folk music and non-Western music that users may be interested in researching or generating harmonies for that don't occur in common time. Additionally, I feel it is worth mentioning that Wu et al. directly favored data types and generating a dataset that is accessible in more commonly available formats and able to be parsed by music21, which supports both casual users and the greater research community.

5.3 A-muze-net: Music generation by composing the harmony based on the generated melody

Goren et al. sought to define a musical model that generates both melody and harmony in a way that is simple and less reliant on massive datasets. They separate their generation into two models, one for each hand of a pianist, and first generate the melody, then the harmony. Their input dataset was a collection of Bach chorales [9]. While I won't go into detail on the training of their model, this paper was worthy of mention due to their unique approach in encoding music and generating it in a way that is lightweight and not so strictly tied to the representations of Western harmony.

For the melody encoding, the authors combined both note and pitch into one value, and instead of considering the names of pitches (i.e. C versus D), they chose to use the MIDI number representation to avoid their method becoming key dependant. The token that actually went into the model was obtained by multiplying the duration of the note with its pitch, making for a total of 1,161 values. To match the harmony with the melody, an encoding is used that matches the number of pitches in the melody that exist within a given chord. Goren et al. found that their model was able to outperform other significant larger and more complicated models in some quantitative metrics and scored the highest in listener examples.

This paper is highly motivating in that it deals with datasets that are much more realistically achievable to use and train for the layperson. The authors note that "... while such models teach us about AI and large-scale pattern extraction, there is little advancement with regard to the foundations of music-making" [9]. A line like this is captivating for me (and maybe even the reader)that, as an end-user (and not a research instutution), competitive goals can be achieved even on smaller datasets that require less computational power.

6 AI Methods in Algorithmic Composition

While I have went into a fair amount of detail with specific papers and described the details of each that I found to be particularly striking, it would also be helpful to point towards a resource that has been particularly helpful for my own work and research. While many papers above describe specific implementations and

their qualitative results, specific implementations may not be the first interest if one is simply looking to create or tune an original algorithm, not for the sake of replication of existing material, but for personal use in composition. Fernandez and Vico provide an excellent paper, *AI Methods in Algorithmic Composition: A Comprehensive Survey* [6] that goes into great detail describing the history, the various methods used to computationally analyze or generate music, and what their specific characteristics is like. If none of the projects above interested you but you're still interested in exploring what algorithmic music could offer towards you individually, this paper could be worth a read to help you find what style of algorithm sounds interesting and applicable towards your individual goals.

7 Conclusion

Machine learning is a field that is exploding in popularity and seeing huge amounts of research. Although its applications in music isn't as widely studied a field as something like natural language processing or image classification, there is a huge breadth of fields that one could start reading or exploring even on a weekend. Whether one wants to play around with modifying the generation models of jazz solos, try to tweak the Bach doodle to generate a different result, or even do some audio processing on their own favorite genres of music, I hope to have provided an interesting compendium of ways that any fan of both music and machine learning could start experimenting on their own. Additionally, using music21 can significantly simplify your own homebrew implementations of generative music (even if not working with known datasets) or empower you to start poking around with the analysis of a large number of composers from the Classical canon.

The field is developing at a rapid pace and there is a variety of choices that one can dig into. I hope I've made you excited for what comes next!

References

[1] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the maestro dataset," *arXiv.org*, 17-Jan-2019. [Online]. Available: https://arxiv.org/abs/1810.12247. [Accessed: 01-Jun-2022].

[2] Henkel, Florian. *"Multi-modal Deep Learning for On-line Music Following in Score Sheet Images."* Institute of Computational Perception.

[3] *History of the Disklavier*. [Online]. Available: https://disklavier.com/history-of-the-disklavier. [Accessed: 01-Jun-2022].

[4] Huang, Cheng-Zhi and Hawthorne, Curtis and Roberts, Adam and Dinculescu, Monica and Wexler, James and Hong, Leon and Howcroft, Jacob. *The Bach Doodle: Approachable music composition with machine learning at scale*.

[5] J. Gillick, K. Tang, R. Miller. *Machine Learning of Jazz Grammars*, Computer Music Journal, 2010.

[6] J. D. Fernandez and F. Vico, "AI methods in Algorithmic composition: A comprehensive survey," *arXiv.org*, 04-Feb-2014. [Online]. Available: https://arxiv.org/abs/1402.0585. [Accessed: 05-Jun-2022].

[7] Kalbag, Vedant and Lerch, Alexander. *Scream Detection in Heavy Metal Music*. [Online]. Available: https://arxiv.org/abs/2205.05580. [Accessed: 02-Jun-2022].

[8] *Music21: A toolkit for computer-aided musicology.* [Online]. Available: https://web.mit.edu/music21/. [Accessed: 01-Jun-2022].

[9] O. Goren, E. Nachmani, and L. Wolf, "A-muze-net: Music generation by composing the harmony based on the generated melody," *arXiv.org*, 25-Nov-2021. [Online]. Available: https://arxiv.org/abs/2111.12986. [Accessed: 03-Jun-2022].

[10] Samuel Castro, Pablo, *Performing Structured Improvisations with pre-trained Deep Learning Models*. https://doi.org/10.48550/arXiv.1904.13285

[11] S. Hakimi, N. Bhonker, R. El-Yaniv, *Bebopnet: Deep Neural Models for Personalized Improvisations*, 21st Int. Society for Music Information Retrieval, 2020.

[12] S. Wu, Y. Yang, Z. Wang, X. Li, and M. Sun, "Melody Harmonization with controllable Harmonic Rhythm," arXiv.org, 21-Dec-2021. [Online]. Available: *https://arxiv.org/abs/2112.11122*. [Accessed: 01-Jun-2022].