Neural Networks to Assist Music Composition

A Capstone Report
presented to the faculty of the
School of Engineering and Applied Science
University of Virginia

by

Conor Monaghan

May 12, 2021

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Conor Monaghan

Capstone advisors: Madhav Marathe, Department of Computer Science, and Worthy N. Martin, Department of Computer Science

Neural Networks to Assist Music Composition

Conor Monaghan
University of Virginia
cm4uw@virginia

ABSTRACT

When writing music, new composers can have trouble expanding their ideas into a cohesive piece. I propose a program that, using Machine Learning and AI, would provide composers suggestions on how to complete parts of a piece based on a database of music. The program would work by analyzing features of a database of music, such as chord progressions, instrument usage, and song structure. Then, given the section of music that the composer has created, the program would suggest continuations of the material, such as contrasting sections or new chords, not writing the song for the composer, but giving the composer potential options on where to take the song. This program could function as a teaching tool for new composers, as well as speeding up the composing process.

1 Introduction

The question of "how do I turn my musical idea into a full song" is one that composers often struggle with. Newer composers, who aren't as proficient in concepts such as song structure and chord progressions, especially have this problem. The main way musicians learn is simply practice and studying music. But it can take years of writing music for composers to learn how to extend their compositions without merely repeating the same material. And if the musician cannot afford to go to music school, the process will take even longer. As a composer myself, I have dozens of unfinished songs that I gave up on because I didn't know how to complete them. After years of composing, my ideas went from fifteen seconds, to thirty, to still just a minute long. A program based in music theory that lets composers learn while they write could drastically speed up the learning process. Even experienced musicians sometimes struggle with expanding songs, and can take a while to find the inspiration to develop their piece. A program that suggests melodic or harmonic material could kickstart that inspiration.

2 Background

Sophisticated music generation has been greatly improved with machine learning and AI. Similar to how text generation works, computer scientists train models that try to predict the next note that will be played in a song. And just like text generation, neural networks and deep learning models have shown to be extremely effective at learning how to create music.

3 Related Work

The application of machine learning to music composition is hardly a new concept; many programs have been written that can take a few notes and extend them into a full song. Wang, Benson, and Sigler created MuseNet, a deep neural network that generates music given prompts such as the start of a melody, a collection of instruments, or a musical genre [1]. Biles made GenJam, a system that can learn to improvise over a chord progression [2]. Adding chords to existing melodic phrases has also been the subject of research. Chuan and Chew created a probabilistic system to assign chords to a melody line [3]. Cunha and Ramalho found that predicting chords partially based on prior sequences in a song led to more accurate results [4].

Although there are many models that can write music or add chords, commercial software that applies this to the composing process is rare. Two somewhat related examples are CAPO and Songsmith. CAPO is an app that attempts to identify the chords in audio to help musicians learn to play that song [5]. It primarily targets guitar players, promising musicians a faster and easier way to learn songs [5]. Songsmith, which cannot currently be purchased, was originally created in 2008 as "MySong" by three Microsoft researchers [6]. Given a sung melody as audio, it generates a chord progression for the user that harmonizes their song [6]. The user can change the "jazz factor" and "happy factor" to influence the type of chords that are chosen [6]. Songsmith seems to be the only tool that specifically assists songwriters with music creation.

4 System Design

4.1 Scope

The goal of the program is to suggest material that the composer can use to complete a musical piece. It is therefore worth considering, what is a song composed of? Since we are focusing on the composition of the music, lyrics will be left to the composer. There are also the different instruments in the piece, the structure of the composition, tempo, and genre. Finally, there is the melody, harmony, and rhythm, all of which are dictated by the notes in the composition. Because this program is

adding material to an existing prototype, not creating a song from scratch, the instruments, tempo, and genre will already be either specified or implied from the composition. To add melody, harmony, and rhythm, the program will suggest notes, whether they be in a single musical line (to create a melody or rhythm), or stacked on top of each other to create harmony. As the program suggests notes to be added to the song, structure will naturally arise as the neural network learns from the existing material. An example of this is how MuseNet can return to melodies throughout its generated songs [1]. Because suggesting notes to create melody, rhythm, and harmony is enough to help with the majority of the composing process, this program will focus on continuing melodies and assigning chords to the song, with rhythm resulting from how the melody is made and where the chords are assigned.

4.2 Input

The program's two primary features, melody continuation and chord assignment, would use a neural network to choose notes/chords that fit the song, so the program needs to have access to the notes of the composition. As there are many popular music composition applications, it makes the most sense to make the program take input through midi data, which almost every music composition program can output. Midi data is also quite low in file size compared to other music formats, and there are multiple libraries made to interpret midi data for machine learning. Since the program would take exported midi files as input, the program would need be separate from the music composition environment, which could be cumbersome for the user to switch between. In the future, the program could potentially be made into a plugin that accessed directly through composition software.

4.3 Architecture

The program would likely be a web-based, client-server implementation. Since the neural network could potentially take a while to add notes or chords, it would be best to have users access it through the cloud. This way, users with slower computers could receive timely results. Since the database and size of the model would also be large, it would also help to store it externally. Finally, if the model needed to be updated, to improve accuracy for example, it would be easier to update the online model rather than needing each user to download new software. The main downside of this implementation is that it would only be accessible online, which would decrease accessibility and usefulness in some situations.

The open-source machine learning library TensorFlow, which is particularly suited for deep neural networks, is often used with Amazon's cloud service Amazon SageMaker to deploy machine-learning models in the cloud. This would be a good fit for the project, since splitting up the training among multiple computers decreases training time. Additionally, using Amazon's cloud services to host the model and program would allow the service to be dynamically scaled up or down as composers increase or decrease their usage, preventing overload as well as unnecessary costs.

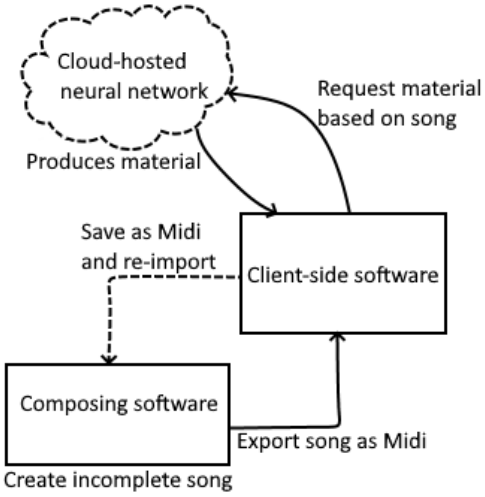

Figure 1: Software Architecture and Process

The final architecture would have client-side software that takes in a midi file, and requests musical material from a neural network model hosted in the cloud, sending the current state of the song to the model. The model then sends back the new material, which is displayed in the software and can be exported to a new midi file.

4.4 Training

As the music will be represented in midi form, an expansive dataset of midi music will be needed for the model to train on. Fortunately, midi datasets are numerous, only a few examples of which are the Musical AI MIDI Dataset, which contains over 77,000 songs [7], and the MAESTRO Dataset, which was used for MuseNet [1] [8].

The neural network would train by trying to predict the next note in a given sequence from a midi file. These predictions would be evaluated by how they compared with the actual next note. As the model develops, the sequences given to the network can be shortened, and the number of notes guessed by the model lengthened. Each note can be thought of as having two main features: the pitch of the note, and the length of the note. The model would try to predict both, with notes closer to the correct pitch and lengths closer to the correct length prioritized. Rests, the periods of silence between notes, can also be thought of as a type of note for note prediction purposes. They simply have no pitch, and can be any length that a note can.

As more composers use the tool to assist with composing, the neural network could further learn by taking into account what suggestions composers build off of and which suggestions composers discard. This could be unreliable, however, since as the tool is separate from the musician's composing application, it would have no way of knowing whether a suggested note or chord was actually used. It could only know which suggestions were immediately rejected for new ones. Perhaps a better way of increasing the accuracy and precision of the model over time would be to update the database with new midis, and tweak the model if certain unwanted patterns are discovered.

4.5 Melody Generation

Midi music is broken up into multiple channels, where each channel has its own sequence of notes, and typically its own instrument. A typical use case for melody generation in this program is to add some length of musical content to the end of a channel. To do this, the model would take various features representing the current state of the song: the pitch and placement of each note for each channel, the chords and harmony being used across channels, the key signature, tempo, and instruments, are all features that would be considered. Due to the somewhat impenetrable nature of deep neural networks, it would be hard to know how each feature actually influenced melody generation. But as the model is trained on a large database of music, these features should all impact the melody in a reasonable way. For example, the instrument for the melody to be written in is important, since each instrument has its own rules in which it can play a melody. Each instrument has a different pitch range, and some instruments can play much faster than others, for example a piano compared to a tuba. Since these examples will be present in the database, they will influence melody generation, and in fact examples of this can be found in the music produced by MuseNet, where the instrument choice can heavily dictate the notes produced by the computer.

Given all these features, and the previous notes on the channel that the model is writing for, the model would predict the next note in the sequence, one at a time. Each time the program predicts a new note, it takes into account the entire song so far, including any notes it has just placed. This means that the song stays coherent as musical content is added. This also means that the program needs something to start with. This would usually be a short, unfinished sequence of notes that the model will expand upon.

Each note has a pitch and a length. Notes can also have individual volumes, but this is less likely to be desired by the composer, as it is generally reserved for certain effects that the composer wishes to include.

An advantage of using this method to generate notes is that the melodic line doesn't actually need to be a melody—the musician could already have the melody completed, but wants a counter line, for instance. Depending on the instrument, previous notes, and the other channels in the song, the program could generate all kinds of lines, from melodies to accompaniment to basslines.

4.6 Chord Generation

Because of this versatility, chord generation is closer to melody generation than one might initially think. If we add in the ability to generate more than one note at a time, there is now a new feature in a "note": the number of different notes played at the same time. Letting the model predict more than one note at a time, where each note is predicted based on the song and any previously predicted notes placed on the same beat, opens up the possibility of chord generation.

This method of chord generation, while effective in theory, may need some support. Since the model is only predicting notes based on features in the song, it is not necessarily creating chords that are understandable. The program may create a line of chords, but the user will not know what those chords actually are. Furthermore, the actual placement of chords may be sporadic and not actually define a chord progression for the song.

To improve on this system, the system should be able to actually predict what the harmony for a certain section of music should be. This is because musicians, especially ones such as guitarists, tend to think of music in terms of chords. Chords play a large part in music theory, and as such, beginner composers have a lot to learn by analyzing the chords of a piece.

The model needs to assign chords to clusters of notes. It can train for this by predicting the chords associated with the melody and accompaniment lines of the midi song files. This is harder than predicting notes, because there needs to be some way of actually knowing what the correct chord for a section of music is. This can be somewhat inferred from the key signature and notes in relation to that key signature, but even then, the key signature must be identified, and the chords will still sometimes be incorrect. The best way to train in identifying chords is through human-labeled datasets. This requires humans to label the correct chords for the midi files, which is a large amount of work. An alternative would be to find musical datasets that already contain chord labeling, but these may be smaller, or not contain as much note data as the midi files. Without this labeling, the model might be able to get close, but will often be slightly inaccurate.

If the model does learn to classify chords, then combining the approach of predicting multiple notes to create a chord, and then classifying that chord, would be

effective. And if the composer has already created some chords before the program is asked to create more, the accuracy would increase further.
Overall, the chord generation is likely the hardest part of this project. Though the program could generate chords without labeling to accompany the melody, unlabeled chords would not be as conducive to educating the composer.

4.7 User Experience

To use the program, a composer would first create a partial song in their favorite music composition application, then export it to midi format. They would then open the client software for the program that allows them to connect to the neural network hosted on cloud servers. After opening their midi file in the software, the program displays a simple sheet music representation of the music contained in the midi file.

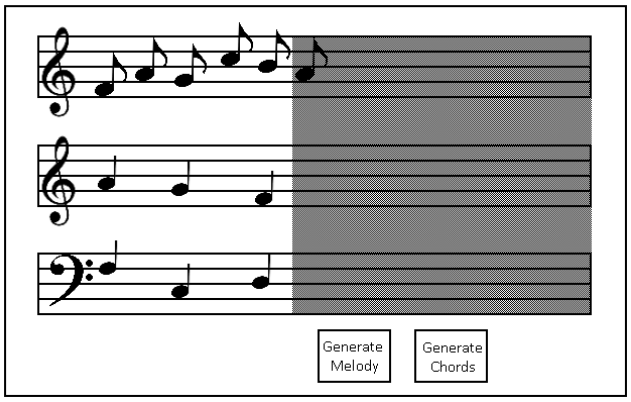

Figure 2: Sample User Interface

The user is able to do some minor editing, such as moving around notes and changing instruments.
To get new melodic material, the composer selects the portion of a channel containing an incomplete melody that is to be completed, and clicks a button. The program then sends a request to the neural network, which takes in the current state of the song, and outputs a sequence of notes to complete the section, which the program converts into sheet music and replaces the selected portion. The user must make a selection; the model needs to have a stopping point to end predicting notes. The model will also produce better material the more material there already is in the song. The user can easily request a new melody in place of the previously-generated one, they just need to press the button again.
For the program the generate chords, the user again has to select a portion of the song. When they press a button to generate chords, the software sends a request to the neural network, which takes in the current state of the song, and outputs a sequence of chords that correspond with the sections selected by the user. The names of the chords are shown below their placement. The user can request new chords to overwrite the old ones.
The software should be made to encourage experimentation and learning, and not laziness. This is part of the reason why the user has to select a specific section to be generated; the whole song cannot be generated at once by the program. This is also why chords are labeled, and why it is easy to replace generated material.
The user can download the changed song into a midi format. The transition to and from midi back into their preferred composition software may lose some information, which is why the program is better as a platform to generate ideas than as an assistant to write the song.

4.8 Monetization

Because of the continual cost of hosting the service in the cloud, and the nature of the neural network being updated and improved over time, a subscription-based payment model makes sense for this project. Users would pay each month to have access to the music generation model. Without it, they could use the client software but it wouldn't be able to generate music. There could be a free trial period, as new users would likely be skeptical if the program could actually produce worthwhile material, and would want to be able to try the program first.

5 Challenges

This proposal has both computational and musical challenges. The model needs to generate material fast enough so that the user can try out several variations in succession without extensive waiting. For this to be achieved, the neural network needs to produce results in a timely fashion, which requires a fast model, in addition to powerful remote servers. The model may take a while to train as well, requiring more time and resources up front.
The musical challenges of the project mostly fall under the model needing to produce acceptable material. This means more time and effort is needed to fine-tune the model. The model also needs to be trained on a variety of music to avoid repetitive or homogenous output.

6 Risks

This project has a high potential risk. It would take a good deal of time and effort to train the neural network model to produce useful melodies and chords, so if the resulting software was not successful, there would be wasted effort. The main reason why the software might not be successful is if the quality of the produced material was simply not enough for composer to make use of it. If composers felt like the software didn't actually help the music composition process, the software would fail. Also, since novice composers are the main target for this software, the price of the subscription could easily turn them away if it was too high.

7 Broader Impact

This tool could help teach and inspire many new composers to pursue careers in composing, but it needs to not let composers be reliant on it. If too many composers because reliant on the software to produce music, they could actually be crippling their skill. The program needs to primarily made to assist with ideas and teaching for musicians who are still learning how to compose. Although this is a tool many composers could use, it needs to not replace human creativity.

There is some algorithmic bias to watch for as well. Since the neural network would be mostly trained on western music, and western music theory, the tool could end up being unsuitable for other styles of music. The program shouldn't make people assume this is the only right way to compose music. Since music from different countries can have very different rules, including different scales and harmony, the program might not work well with all types of music.

8 Conclusions

AI-assisted composing is currently an untapped market. There is great potential for computers to spark the creativity of humans and expose patterns that no one had seen before. A program that suggests continuation of musical phrases and chord progressions could progress the careers of many composers. But although there is great potential, there is also possibility of failure or even harm to new composer's creativity, if the program is not made with the right goals in mind. If it is, however, it could break new ground in machine learning music.

9 Future Work

Future work on this project would include an implementation of all that was mentioned so far. New features would likely include a more useful midi editor; although they exist, users would likely want to edit suggested material and use the client-side software in more ways. Another feature could be the ability to suggest instruments that the composer could use in their piece. Finally, the program could be made into a plugin that works within a music composition application, allowing machine learning music generation within a typical workflow.

REFERENCES

[1] Justin Jay Wang, Nicholas Benson, and Eric Sigler, 2019. MuseNet. OpenAI. https://openai.com/blog/musenet/

[2] John A. Biles, 1994. GenJam: A Genetic Algorithm for Generating Jazz Solos. In *Conference: International Computer Music Conference (ICMC), (*July 1994), 131-137. https://www.researchgate.net/publication/2342018_GenJam_A_Genetic_Algorithm_for_Generating_Jazz_Solos

[3] Ching-Hua Chuan, Eliane Chew, 2007. A hybrid system for automatic generation of style-specific accompaniment. https://www.researchgate.net/publication/228698591_A_hybrid_system_for_automatic_generation_of_style-specific_accompaniment

[4] Uraquitan Sidney Cunha, Geber Ramalho, 1999. An Intelligent Hybrid Model for Chord Prediction. In *Organised Sound* 4(2), 115-119. DOI: 10.1.1.68.6047

[5] CAPO. SuperMegaUltraGroovy. http://supermegaultragroovy.com/products/capo/

[6] Ian Simon, Dan Morris, Sumit Basu, 2008. MySong: Automatic Accompaniment Generation for Vocal Melodies. In *ACM CHI* 2008 (the 26th SIGCHI Conference on Human Factors in Computing Systems), p725-724. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/mysongchi2008.pdf

[7] Yifei Teng, An Zhao, Camille Goudeseune, 2016. MIDI Dataset. https://composing.ai/dataset

[8] Curtis Hawthorne, et al., 2019. Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset. In arXiv:1810.12247v5.