WPI

Chord Craft:
AI Chord Generator

A Major Qualifying Project WORCESTER POLYTECHNIC INSTITUTE
A Major Qualifying Project report submitted to
the faculty of Worcester Polytechnic Institute
in partial fulfillment of the Degree of Bachelor of Science

Project Team:
Sameer Desai
Thomas Kneeland
Darren Kwee
Ilana Whittaker

Project Advisors:
Professor Scott Barton
Professor Charles Roberts

Submitted to the Faculty of the Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science.

Abstract

The creative process in music composition can present challenges for musicians, prompting the exploration of online tools to overcome these barriers. This project introduces ChordCraft, a software-based solution to assist musicians in generating chord progressions. By utilizing the JUCE C++ framework and a Hidden Markov Model, ChordCraft offers the ability to input a MIDI file and receive a corresponding MIDI chord progression. We employed quantitative evaluation metrics to assess the model's performance, in addition to user studies to gather qualitative feedback. ChordCraft aims to enable musicians of all levels to compose, offering a toolset to encourage musical expression.

Acknowledgements

We would like to thank our advisors, Professor Scott Barton and Professor Charles Roberts, for their constant and involved effort with our project. We would also like to thank everyone who provided feedback for our software.

Table of Contents

Table of Figures

Table of Tables

Executive Summary

Many people find themselves connected to music, whether that be through listening, performing, arranging, or composing. One problem that many musicians face is learning how to overcome creativity barriers, especially when composing. There are many strategies to break these barriers, including the use of technology. Numerous websites and applications aim to help musicians with the composition process, but often fall short by being expensive, ineffective, or difficult to use.

The purpose of this project was to design and implement an intuitive and technologically-based solution that musicians can use as an aid in the composition process. After conducting research of past projects and reading academic articles, we ultimately decided that a chord generator would be a suitable solution. Our generator, called ChordCraft, is a VST (Virtual Studio Technology) that users can access through DAWs (Digital Audio Workstations) like Garageband and Ableton. The goal of this software is to allow users to input a MIDI melody file, and output a MIDI chord progression to accompany the melody.

To begin creating this software, we needed to determine the most suitable framework to use for coding. After much research, we decided to use JUCE, a C++ framework commonly used to create musical plugins. The front-end design was created using JUCE. To accompany the front-end of the software, we implemented an HMM (Hidden Markov Model) to generate the chords in the back end. This model uses a dataset consisting of hundreds of songs to predict states represented by chords. The back-end also includes a voice leading algorithm which aims to retain common tones and limit jumps between notes in chords. To bridge together the back-end and front-end of the software, we implemented a REST API using a Python server. We made this decision as it ensured that we could still make updates to the back-end without affecting the status of the front-end.

To evaluate the model's output, we established quantitative metrics to check its adherence to musical rules. Our metrics checked for diatonicism, note fitting, and common tones. Diatonicism measures the percentage of chord tones within a given key. Note fitting checks for how well the melody fits with the chords, which is determined by the percentage of melody notes

that are not an avoid note under their given chord (avoid notes are notes that would cause dissonance or tension). The last metric, common tones, checks for the frequency of shared notes between consecutive chords. These metrics, which are rooted in common music theory principles, provide an extensive evaluation framework for our model's performance.

At the conclusion of the evaluation period, we conducted studies with members of the WPI community to gain feedback on UI features, as well as the success of the generated chord progressions. Each participant completed a guided tutorial of ChordCraft, being asked to generate chord progressions while exploring the different parameter options. We encouraged each participant to share their thoughts about ChordCraft during the study, which helped us determine areas of the software that needed the most improvement.

After completing the methods associated with this project, we formulated several recommendations for how to improve ChordCraft for the future. This included determining new parameters that would be helpful, reformatting UI features, and increasing the complexity of chord possibilities in the model.

1.0 Introduction

Music composition is a concept that is familiar to various levels of musicians. For some, it's a career, and for others, it's a hobby or interest. Like any other creative outlet, music composition can leave individuals struggling to come up with new ideas for their pieces. Parts of a composition include melody, harmony, rhythm, voice leading, and more. The earliest records of music trace back to pure rhythm, with people using their hands and knuckles to create sounds (Stanford, 1930). As composition gained popularity over time and research evolved, notes and chords made their way into rhythmic music pieces (Juchniewicz & Silverman, 2013).

A successful chord progression supports the melody of the piece, and provides moments of tension and resolution. A chord progression can also convey a piece's emotion, depending on the order of chords and use of non-diatonic chords to create more dissonant moments (Juchniewicz & Silverman, 2013). Certain chords within the progression play a specific role, like tonic (the root chord), dominant (the fifth chord), or subdominant (the fourth chord). The use of these chords can create cadences within progressions, which establish moments of conclusion.

Over time, software has played an increasing role in the production and composition of music. The early 20th century introduced musicians to recording, which enabled them to hear themselves play. In the mid-20th century, the use of synthesizers became apparent and as time passed, these musical advancements began to transition from hardware to software (Peters, 1992).

One resource that combines music and software is Virtual Studio Technology (VST). A VST is a software interface standard that users of programs like Garageband or Logic Pro can download and use as a plugin. VSTs can assist users with audio effects, electric instruments, auto tuning, and more (One Revolution People's Music, n.d.). VSTs come in a variety of skill levels and price, so musicians with varying levels of experience find themselves using VSTs in their everyday lives.

Scientific models can predict states in sequences of varying scenarios. The Hidden Markov Model is a statistical model that uses hidden states, and models transitions between states based on probabilities from data sets (Daniel & Martin, 2023). HMMs can help predict

chords by utilizing these hidden states and analyzing the most likely chords to follow previous chords (Fu & Lin, 2019). This capability makes HMMs a fine choice to develop a chord progression generator VST.

The purpose of this project is to create a software that musicians can use to automatically generate chord progressions for a given melody. To complete this project, we created a Python-based HMM to generate chords for the melody. We also set up a server to host the Python code in JUCE, which tied together the back-end Python development to the front end JUCE design. At the intersection of music and technology, this project aims to provide musicians with an accessible solution to creating chord progressions for various melodies.

2.0 Background

Musical creation is at an all-time high, thanks to advances in technology and resource accessibility (Lee & Lee, 2018). Because of the increased popularity, developers have recognized the demand for software that helps musicians with their writing process (Mortimer et al., 2014). However, this isn't a new concept; the first instances of using computers to create music trace back to the 1950s (Peters, 1992). The development of common algorithms since then allowed for more complex and unique harmonies to be created (Langston, 1988). There has also been an increase of artificial intelligence being used to serve the same purpose, allowing for automatic composition, remixing, and songwriting (Chu et al., 2022). In addition, the need for programs that automatically apply voice leading to chords has also been on the rise (Mortimer et al., 2014). Many of these softwares use machine learning techniques to implement established music theory rules and analyze patterns.

How can musicians use technology to help them write music? Before considering this, it's important to discuss common music theory practices like melody, harmony and voice leading. It's also necessary to explore the ethical implications behind the creation and use of these software tools.

2.1 Introduction to Songwriting

For the purpose of this paper, we define a song as having a main melody accompanied by harmonies and chords, recognizing that this definition may not encompass all musical practices but aligns with our research interests. Within this framework, it's notable that most forms of popular music utilize this structure.

2.1.1 Melody

A melody is often a song's distinguishing characteristic, which Merriam-Webster defines as "a sweet or agreeable succession or arrangement of sounds". Two foundational concepts of melody composition include pitch and rhythm. Once a composer has an idea for a melody, a logical next step in the composition process is to write additional voices that can create some kind of harmonic accompaniment to the melody. In conventional Western music, this can take

many forms - whether that be a guitar, piano, or even a full ensemble like a string orchestra or choir. Harmonic structure is commonly defined by which chords are used, relative to the key of the piece (Bharucha & Krumhansl, 1983). These chords have a specific relationship with the notes in the melody which helps to give musical context to each note, depending on the pattern of tension and release throughout the progression. However, this relationship between chords and notes does not exist as a one-to-one mapping; that is, for any given melody note, there are several possibilities for different chords that may work well. There are many factors which may influence which chord the composer chooses, such as the overall style of the music or the harmonic function of that chord.

2.1.2 Harmony

There are many situations in which a composer will write a harmony to accompany the melody they just wrote. Classical harmony often adheres to principles established during the common practice period of Western art music. This practice relies on the diatonic scale, establishing tertian chords which are built on stacked thirds. The use of the diatonic scale forms the basis of harmonic function, where notes and chords in the scale serve different roles, like tonic, dominant, and subdominant. As an example, classical and jazz genres will not follow all the same rules and styles. Jazz theory differs from classical music, due to jazz chords using more extensions, and also the introduction of instrumental improvisation (Bharucha & Krumhansl, 1983). Common Practice Period (CPP) western art music, on the other hand, follows some unwritten rules established by well-known composers of the 18th century, like avoiding tritones, which are harmonic intervals of three whole steps. When writing harmonies for singers, musicians must take into consideration the ranges for each of the four parts (soprano, alto, tenor, and bass). A well-written harmony will bring out and support the melody, but not overpower or clash with it.

Another fundamental element of music theory and songwriting is creating progressions of chords. Chords are often considered to be three or more pitch classes played at the same time to create a full sound, and they usually support a melody on top of them. Chords are imperative to the flow of any song, and give it a sense of structure and feeling (Bharucha & Krumhansl, 1983). A triad chord contains three unique notes: the bottom note of the chord is called the root, followed by the third and the fifth. Four common categories of triad chords are major, minor,

augmented, and diminished. Figure 2.1 shows the corresponding sheet music notation for the four chords with F as the root note, and it shows the different intervals that are unique to each chord type. There are also 7th chords, which are just like the previously-mentioned chords except with the 7th scale degree included in the chord, which results in four independent notes being played at the same time.

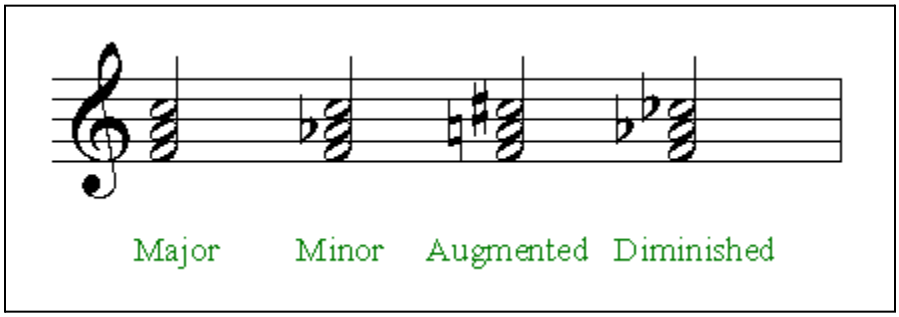


Figure 2.1: *Image of the Four Major Triad Types*

It's important to study the relationships between chords, and the various moods that selecting chords in a specific order convey. A chord progression is a series of chords that represent harmonic changes of music with understandable notations (Absolu, 2010). Chord progressions play different roles depending on the genre of music. For example, in jazz, a predetermined chord progression often plays in the background while the instrumentalist improvises (Absolu, 2010). Figure 2.2 below shows the most commonly-used chord progression patterns, closely following common music theory practices (Mastering.com).

Starting Chord	Usually followed by...	Sometimes followed by...	Rarely followed by...
I	IV or V	vi	ii or iii
ii	V	IV or vi	I or iii
iii	vi	IV	I, ii, or V
IV	V	I or ii	iii or vi
V	I	IV or vi	ii or iii
vi	ii or V	iii or IV	I
vii°	I or iii	vi	ii, IV, or V

Figure 2.2: *Chart of Common Chord Progressions*

Many of these music theory and composition concepts can be thought of as not hard-and-fast rules, but rather guidelines for musicians to follow which will produce something that sounds similar to what has been created in the past. Similar to color theory in visual arts, "breaking" the rules can still create something pleasing, iconic, or serve as an effective storytelling element in a work of art. Because of the many complex ideas and decisions that composers work with, many software tools have been developed to aid the composition process.

2.1.3 Voice Leading

Voice leading is a fundamental concept in music composition, as it dictates movement between parts within a musical texture. Proper voice leading is essential for music composition, which helps to determine proper chords for a song. Once a chord progression has been determined, the composer then needs to figure out how these chords are going to be voiced, so that they transition smoothly from one chord to the next.

Other rules include avoiding parallel motion, which is when two parts move together in the same intervals. The best practice is to use contrary motion, which does the opposite. For example, the soprano part may move down an interval while the alto interval moves up (Pachet & Roy, 2001). Another common rule is to keep common tones in the same voice part from chord to chord, which helps with singability and ease of reading. Achieving successful voice leading

can enhance the harmonic richness of a piece, and create a sense of tone color. Crafting the individual lines of a chord progression can be a tedious task but is a foundational skill among composers, arrangers, and those with an interest in music.

2.2 Music Encoding Formats

While these musical ideas exist in the minds of composers and musicians, it can be useful to encode them so that they can be shared with others. Traditional notation offers a comprehensive system, including rhythm, pitch, and dynamics. Modern genres like jazz and pop sometimes utilize lead sheet notation, which condenses the traditional notation, giving the musician more freedom to make choices not written in the music. Musical Instrument Digital Interface (MIDI) has also increased in popularity in music production, offering a standardized protocol for the communication between computers and sheet music. This section explores the different formats that musicians commonly utilize.

2.2.1 Traditional Notation

There are many forms of music notation which can be used to preserve all of these musical ideas onto paper, allowing composers to share their musical ideas with musicians. The most comprehensive traditional music notation includes all of the musical ideas of a piece, including the exact rhythm, pitches, dynamics, articulations, lyrics and expressive markings for all of the instruments and voices involved.

2.2.2 Lead Sheet Notation

For more modern genres such as jazz and pop, there is a condensed notation called “lead sheets” which offer a reduced set of musical elements - namely just the melody rhythms and pitches, lyrics, and chord symbols. This is typically done to give musicians room for improvisation, which allows them to put their own “spin” on various elements which aren’t defined in this notation, such as the voicing of chords or the rhythm of the bass line. This can also be used as a shorthand version of songs, which contains minimal musical elements required to define most of the musical ideas (Lead Sheet | Berklee College of Music, n.d.). Figure 2.3 outlines a diagram of a lead sheet, displaying the chord symbols and lyrics.

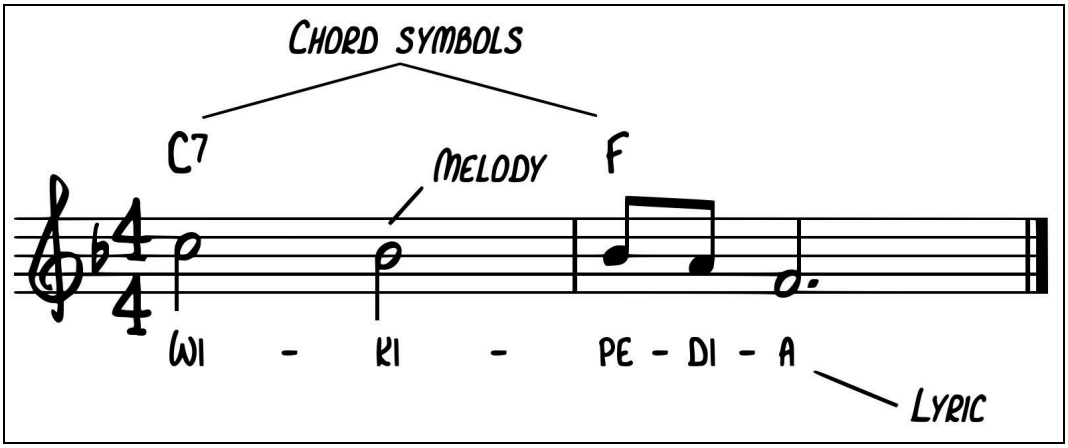


Figure 2.3: *Lead Sheet Diagram*

2.2.3 Musical Instrument Digital Interface (MIDI)

Musical Instrument Digital Interface is a standardized protocol designed for electronic musical instruments to connect with computers. Rather than containing audio signals, MIDI consists of instructions on how music notes should be played, including the length, velocity, pitch, and more. Since MIDI is designed for live performance, it consists of a series of digital messages which can be stored in a file sequentially or input live from an instrument (Gibson, n.d.). MIDI can be used to encode the majority of music notation, and is a widely accepted standard for storing songs in digital format. There are countless music softwares that utilize music in MIDI format. For this project, we explored softwares that aid in the music composition process.

2.3 Composition Software

The following section explores several software features which aid the composition process by analyzing their utility for composers, implementation algorithms, and existing software. These features help composers generate or realize their ideas - such as an “instrument” which helps realize a specific chord, a collection of preset chord progressions based on genre, or even the automatic generation of chords given a melody as input. To begin, here are a few key concepts and definitions which are important to understand in the context of composition software.

2.3.1 Chord Building Tools

There is a fair selection of tools that aim to simplify the process of manually writing out chords. This could be useful for someone who is a novice writer, or even for more experienced producers that are looking to streamline their workflow. One such tool is ChordJam, which prompts users to input a chord symbol, then customize the number of voices in that chord and the octave range for each voice (CHORDJAM by Audiomodern™ | The Ultimate Chord Machine). Chord Player also works in a similar fashion. Another tool worth highlighting is the Nordmann Chord Generator, specifically designed as a Max For Live MIDI effect device for Ableton Live, which section 2.4.1 further discusses. It helps the user create a chord from a single MIDI note, with that triggered note setting the chord's root (Nordmann, 2011). Users can create major, minor, diminished, and augmented triads and add up to two additional chord tones as shown with the buttons on the left side of Figure 2.4.

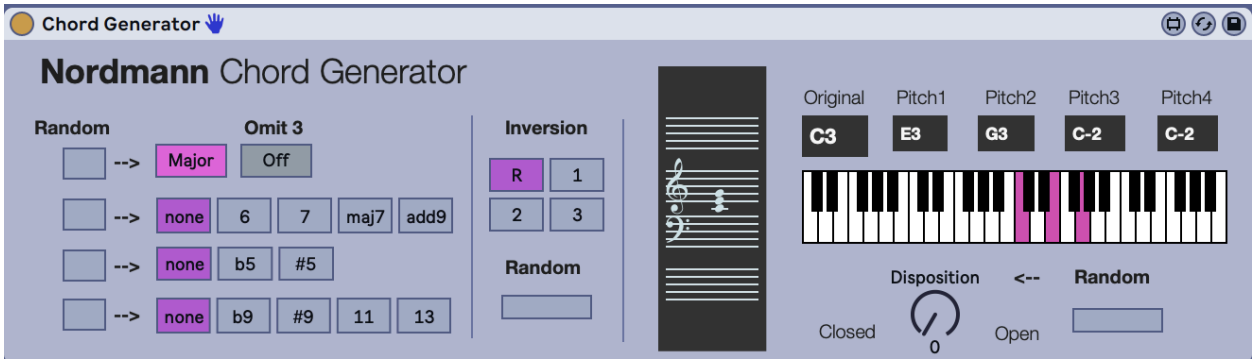


Figure 2.4: *Nordmann Chord Generator UI*

Users can also modify chord inversions, choosing from Root position, 1st, 2nd, or 3rd inversion. Additionally, the disposition knob, with a range from 0 to 4, allows for the expansion of notes across multiple octaves, from closed to open voicings. Every parameter can be randomized to easily generate unique chords just from one note. Nordmann Chord Generator has a simple user interface which allows a user to create complex chords with five voices just by triggering one MIDI note.

Overall, while these types of tools are considerably less advanced than some others, they still prove to be a practical help to producers. Most of these tools also thrive off of having a clean user interface, which is crucial for efficient use. The next level of software composition tools covered are able to create entire chord progression from a melody.

2.3.2 Automatic Chord Progressions

There are several tools which can automatically choose chords or a chord progression considering the melodic context of a phrase. This type of tool can be useful in many cases, such as for composers who may not have the technical music theory background to produce a chord progression from scratch or for those seeking inspiration regardless of their experience level. This section will explore software that uses deterministic models, probabilistic models, and discuss the implications of deep learning on generating chord progressions. The simplest form of this software tool only considers the context of the melody's key, either randomly generating a chord progression or selecting from a collection of presets.

2.3.2.a Deterministic Models

Chord Progression Generator, a simple web application created by SoundGrail, will consider a key, musical mode, and tempo to generate a four-bar chord progression which can be exported to MIDI (Chord Progression Generator, 2023). Additionally, the user may select whether or not they wish to include 7th chords in the generation. While there may be some story-telling advantage to having an element of randomness in a piece of music, the chord progressions that are generated randomly generally do not follow any music theory conventions and are therefore less useful to composers.

Some tools allow the user to choose from a collection of chord progression presets which do follow standard music theory conventions and therefore may be a more logical place for a composer to start. Scaler 2 is one example of this - a VST which has a collection of over 600 chord progression presets, based on various genres, popular songs and those curated by artists (scalerplugin.com). These presets can be voiced then manually edited through their intuitive UI, as shown in Figure 2.5. This tool also presents all of the conventionally-used chords within a key to the user, allowing them to easily voice key-appropriate chords. There are also a few other notable features, such as recommending chord substitutions for individual chords in a

progression and a modulation guide, which will suggest a series of chords which will help modulate into a specified key. While Scaler 2 does allow the user to input a melody as a MIDI file, the only context which the program considers is the key.

Figure 2.5: *Screenshot of Scaler 2's Chord Progression Presets*

2.3.2.b Probability-Based Models

To consider the context of other chords within the progression and the melody notes, probability-based modeling techniques, such as a Hidden Markov Model, are used. When composing a chord progression, there are two main factors to consider - the first is the melody notes which will be accompanied by the chord and the chords surrounding them. A HMM can consider all of this context and will produce the progression which based on the examples it was trained on, is most likely to occur.

Three examples of software will be discussed here: Songsmith, Hookpad and Vielklang. While Songsmith is the only application which has extensively publicly-available documentation on its algorithm, Hookpad and Vielklang provide similar functionality and results.

A Hidden Markov Model is built on a Markov chain, which is a model that records the probabilities of how likely a specific state is to occur based exclusively on its previous state (Daniel & Martin, 2023). Each state is contained within a node, and each node in the model has a relationship with every other node (including itself), storing the probability of moving from that node to any other, which Figure 2.6 visually represents. Here, each node represents a chord, and the probabilities of moving between nodes represent the likelihood that a chord will transition to another chord. A HMM builds on this by considering the Markov chain as a series of unobservable states, which in this case represents a series of unknowns, that each have a probability of transitioning into a different series of observable states - which are melody notes in this case (Daniel & Martin, 2023). These probabilities represent the likelihood that a specific chord is played alongside a specific note. However, since this relationship is not one-to-one, there are several nodes in the Markov chain which may correspond to observable states - in the same way that there are several chords that may accompany each note and several notes that may be supported by the same chord. Using the Viterbi algorithm, which is "a recursive optimal solution to the problem of estimating the state sequence of a discrete-time finite-state Markov process" (Forney, 1973), the most likely sequence of unobservable states can be predicted based on a sequence of observable states.

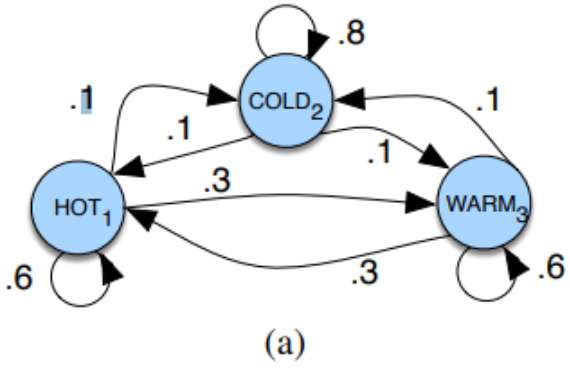


Figure 2.6: *Markov Chain of Weather States*

One piece of software that uses this algorithm is Songsmith (Songsmith, n.d.), a standalone application developed by Microsoft that generates chords from a melody given as audio input. First, a user sets the tempo of the song and chooses one out of 31 styles for the virtual instrument that performs the accompaniment track (Basu et al., 2014). Then, the user records vocals or plays an instrument into the computer mic with a drum track as background to stay in time. At this time, the application only supports a 4/4 time signature. After the user finishes recording, Songsmith will then generate a chord progression that it determines to fit with the melody. By default, a harmonic rhythm of one chord per bar is used, although that can be additionally adjusted to either two chords per bar or two bars per chord. Figure 2.7 below illustrates Songsmith's UI design.

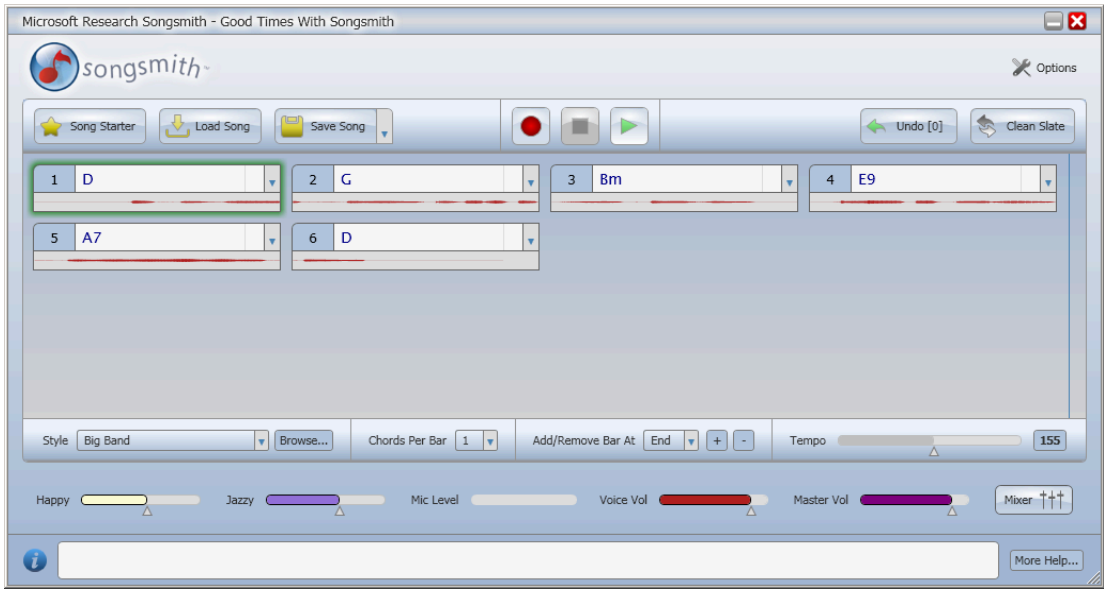


Figure 2.7: *Songsmith's User Interface*

Songsmith uses a HMM which was trained on data from 300 lead sheets. A Markov chain was built by calculating the probabilities of each node, representing a chord, moving to another node. The probabilities from moving from the unobservable states in the Markov chain to the observable states was calculated by the likelihood of each chord corresponding to any pitch class. Each of the leads sheets were transposed into the same key and all calculations were made relative to that key (Morris, 2008).

One key feature of Songsmith which separates it from other software such as Hookpad is that it allows the user to adjust parameters which influence the chord progression generation. There are two parameters that can be changed on a discrete scale: “happy” factor and the “jazz” factor. The “happy” factor refers to the major / minor tonality of the chord generated. For Songsmith’s HMM, two models were created, and therefore two probability matrices - one that represents the probabilities of minor tonality songs and the other for major mode. Depending on the value of the happy factor, the weight of each transition matrix will differ and therefore alter the chord which is generated based on the likelihood that it comes from that key or its relative major / minor. The “jazz” factor refers to how much consideration is given to the melody vs. the chord transitions, where a low value ignores the melodic notes and instead generates purely based on chord transition probabilities.

Another example of software that takes advantage of a similar probability-based model is Hookpad. While the exact algorithm used is unknown, their marketing which claims to use “AI” and output results which are similar to Songsmith suggest a HMM. Hookpad is a standalone software that helps users compose chords from a MIDI melody. After the user completes their project, the software outputs MIDI files, which the user can then drag into any DAW for instant use. HookPad has a colorful interface where the user can create a melody using notes in a specified major scale. Figure 2.8 displays the interface:

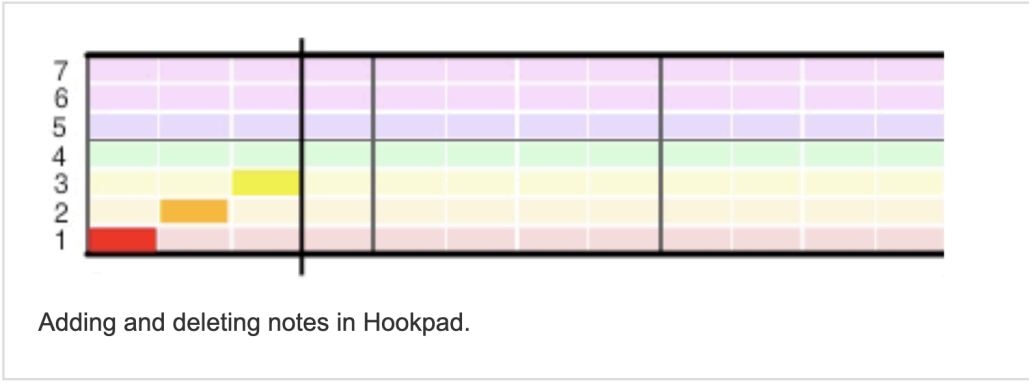


Figure 2.8: *Hookpad’s User Interface*

The numbers 1-7 correspond to the 7 notes of each major scale. To enter a melody, the user clicks through the grid, selecting different notes. Using the up and down arrows, the user

can also change the octave of the input notes. The user can also click and drag to change the duration of each note, enabling the creation of rhythmically complex melodies (Julia et al., 2021). There are also keyboard shortcuts to apply accidentals to notes in the melody for those notes outside of the scale. Hookpad also contains a setting that supports up to 4 independent melodies to be inputted into the same file. This is useful for generating chords for larger ensembles. Figure 2.9 shows an example of the chord interface that Hookpad contains:

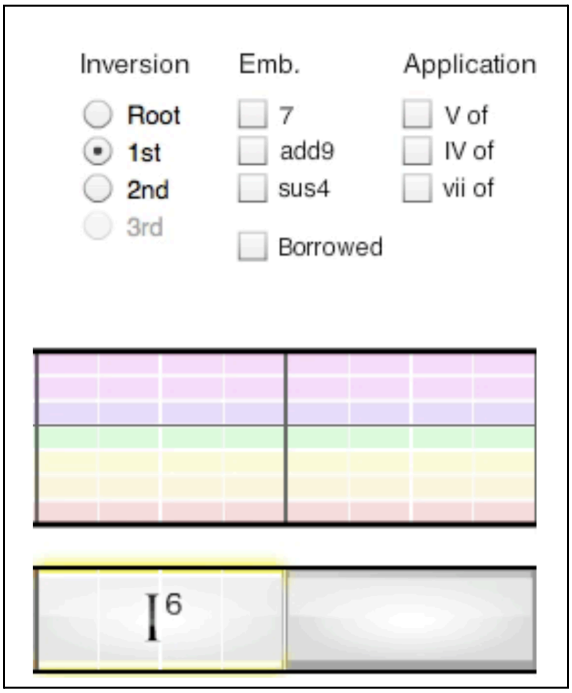


Figure 2.9: *Hookpad's Chord Interface*

There are several parameters, including inversions, extensions, applications, and borrowed. Borrowed chords incorporate chords from parallel keys, and applications are modifying notes in a chord usually by lowering or raising them by a semitone. Besides allowing the user to manually input musical notes and chords, Hookpad also has automatic composition features. The first feature is called "Magic Chord", which selects chords that traditionally sound good together. Hookpad contains a large library of previously-used successful chord progressions used in popular songs. The library, called the TheoryTab database, contains 44,320 songs. The user asks Hookpad what chord would work best in a specific situation, taking into account

chords that were previously added. Hookpad then searches through its library to find similar chord progressions, and then provides an educated suggestion to the user. Hookpad also supports different instrument sounds. After the user completes their project, they have multiple different ways of exporting their file. These options include MIDI, MP3, score, leadsheet, etc.

Hookpad's popularity began to rise during the pandemic when educators and musicians were looking for innovative and contactless ways to teach music. This software is geared towards experienced musicians due to the complexity of the options and parameters offered to the user. Despite that, it has proven itself to be a useful and accessible tool for less experienced musicians looking for assistance with composition and songwriting.

Vielklang is a harmonizer VST plugin which, given a melody via audio or MIDI file, is able to generate harmony with two to four voices using different harmonization modes. Its main use is for creating vocal harmonies with a choir type effect, but when using a MIDI file as the input it can be used as a chord progression generator.

First the user drags in an audio or MIDI file and Vielklang auto detects the key of the melody. Then the user clicks harmonize and chooses from seven different harmonization modes (see Harmony Settings in Figure 2.10). Vielklang will display the chords generated in the piano roll with the melody being highlighted. The user is able to change the harmony mode which changes which chords are chosen and they can choose different chord voicings for the progression (Zplane, 2012, p. 26). There is no documentation on the algorithm behind Vielklang. The tool's notable features include the ability to generate chords from a MIDI melody, multiple harmonization modes for chord alterations, chord voicing options, and the piano roll view which shows the melody and generated chords together.

Figure 2.10: *Vielklang's User Interface*

2.3.2.c Other Deep Learning Methods

Besides the approaches described above, there have been several others in the deep learning realm that are promising in generating useful chord progressions. While a HMM can be considered an unsupervised model, they are mostly trained in a supervised manner, which many scientists classify as "semi-supervised (Tamposis, 2019)." In the context of chord progressions and melodies, this means that "appropriate" relationships between each chord node and melody note node are defined in the dataset by which the model is trained on.

There has been lots of work done in the realm of generative music using recurrent neural networks (RNN), generative adversarial network (GAN) and transformers, both with feature extraction and without (Ji, 2020).

2.3.3 Voice Leading

Besides chord choice, another important consideration to the sound of the music is how the voices of the chord transition to the next. Automated chord voicing tools are focused not on original chord generation, but instead the process of realizing each chord into pitches, and figuring out how those pitches should be arranged in a fashion that facilitates smooth movement from one chord to the next. The process of enforcing traditional voice leading rules can be modeled with constraint programming, which is "identifying feasible solutions out of a very large set of candidates, where the problem can be modeled in terms of arbitrary constraints" (Constraint Optimization, 2023).

SATB Part Writer is an exemplar voicing tool, taking in any chord progression provided by the user, and realizing it into an exportable four-voice arrangement. It follows the western European classical tradition of voice leading, treating each of the "rules" and conventions as constraints. These include utilizing contrary motion between the bass part and the rest of the parts, avoiding parallel fifths octaves, and doubling the root of the chord. Another similarly designed application is Harmony, an open- source application. There's also Gregh Merrill's Voice Leading tool, which intakes an XML file and gives the user an evaluation on how closely their own voice leading follows classical conventions. It checks the vocal ranges of each part, spacing between voices, parallel fifths and octaves, voice crossing and other fundamental music theory rules (*How's My Voice Leading?*, n.d.). This is a useful tool for musicians who may be unfamiliar with these rules but want to elevate their compositions or arrangements.

2.3.4 Good User Interface Practices

It is essential for a platform to have a user interface that provides ease of use without sacrificing functionality. While most VSTs have some sort of user manual, relying on users to read it in order for them to have any basic understanding of the VST's functionality is not ideal. With a proper interface, users should be able to infer at least the core functionality upon first glance. Three principles can be followed to achieve this: organize, economize, and communicate (Martin, 1999).

To organize is to "provide the user with a clear and consistent conceptual structure" (Martin, 1999). In terms of consistency, functionally similar elements should be grouped together

and given similar input forms. For example, buttons should follow a uniform shape with one another unless their functions are distinctly unique from others. There's also external consistency, which dictates that an interface should adhere to commonplace conventions that appear across most applications. An example of this would be that most word processors share the same symbols for bolding or italics, or that a cyclical arrow is used as a reload button in most web browsers. Lastly there is also real world consistency, such as a red octagon being a universal symbol for halting. Screen layout is also important- in most cases, it is good practice to follow a gridlike form, and elements should be sized based on their importance. Economizing is about accomplishing the most using the least amount of elements possible. Redundancies should be avoided. For example, most DAWs do not have separate pause and play buttons; one button is used to start and stop play. Communication mainly emphasizes readability. This includes choosing appropriate font styles and sizes, and clear color schemes.

All three of these principles share some interplay with one another, and when utilized correctly, lead to an easily deciphered application. One example which applies all of these concepts is Drummer, which is described in the next section.

2.3.4.a Drummer

Drummer (*Get started with Drummer in Logic Pro for Mac*) is a VST that is exclusively available in Logic Pro, with Garageband having a limited version. It is a great example of a user interface that is easy to understand, yet versatile. There are 33 different styles of drumming to pick from, each style being represented by a drummer character. Most popular genres, from funk to Latin to EDM, are played by one of these characters. Each drummer character also has a selection of beat presets to choose from, and those variations can be further modified by the user in some of the following ways. Figure 2.11 below displays the Drummer user interface.

The user can change the dynamic of the beat, which doesn't just change the volume, but alters the drummer's striking technique and overall aggression. The beat's complexity can also be changed, adding in quicker sequences of notes or more layered syncopation as the complexity rises. The dynamic and complexity are both controlled via an XY pad, allowing for micro adjustments in any direction, with dynamics assigned to the Y axis and complexity on the X. It

allows the user to intuitively modify both of these parameters at the same time. The user may also alter the drumset's arrangement, such as toggling the toms, hi-hat, or ride cymbal on and off, and can adjust the frequency of drum fills or the swing levels. Lastly, the hi-hat pattern and the kick/snare pattern can be selected from a few different presets.

In addition to these presets and user-defined options, the user can instead choose to have the drummer listen to another track, and place the kick and snare accordingly to what it observes. This allows the drummer to be rhythmically locked in with the other instruments in a given project, no matter how many other drumming parameters the user chooses to modify.

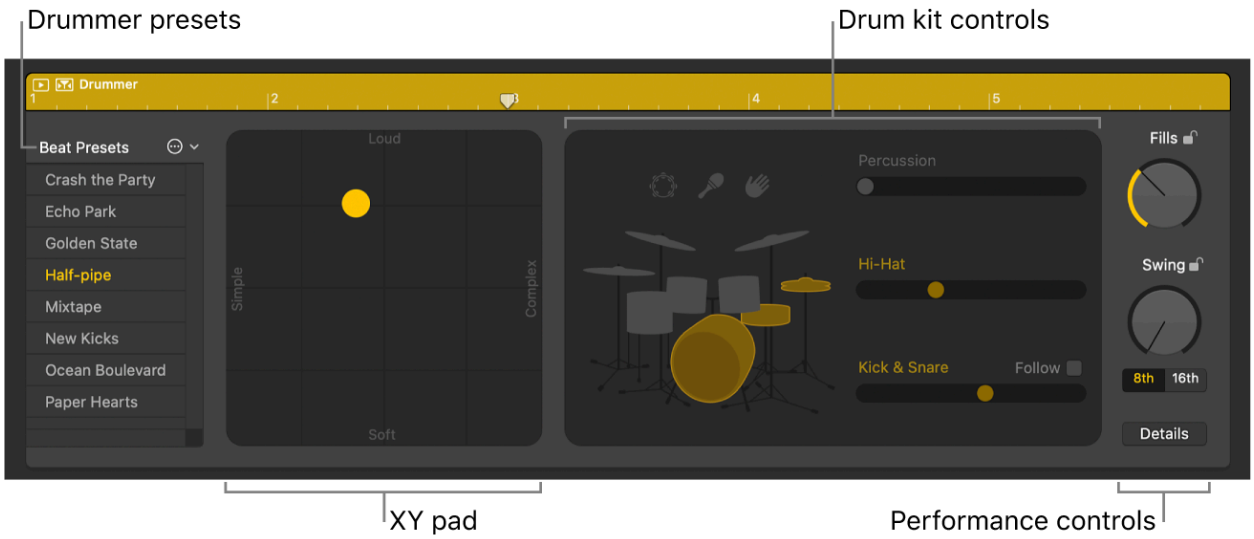


Figure 2.11: *Drummer's User Interface*

Lessons can be learned from Drummer's user interface. It manages to neither alienate new users nor bore experienced producers (Drummer in GarageBand, n.d.). Someone can pick it up for the first time without having to read any instructions, and yet the number of ways it can be configured are virtually endless (Drummer in GarageBand, n.d.). It follows organization, economization, and communication effectively. The gridlike layout is well-constructed, with appropriately sized elements. All of the symbols are easily digestible, the text is clean, and the color scheme is easy on the eyes and effective at showing what is selected. The XY pad is a brilliant example of economization. Drummer provides an enticing blueprint for how to

effectively lay out a practical generative MIDI VST. There are also direct parallels between Drummer and our project, since Drummer has that ability to use another track as reference for its own generation.

The only drawback to Drummer is its limited DAW availability, and lack of developer or community documentation. The actual drumming itself is also not of much interest to our project, since there is no element of pitch, but the form factor and ease of use is exemplary.

2.3.4.b ChordPrism

We also explored ChordPrism, a VST plugin designed to generate and visualize chords through a piano-like interface. While the VST doesn't directly produce audio output, it offers the ability to export MIDI files to the user's DAW for playback on instrument tracks (*HOME - Chordprism*, n.d.). It's compatible with a wide range of DAWS, and features a wide variety of customizable parameters including options for octave, velocity presets and chord groups, all of which significantly impact the resulting chord progression's characteristics and sound.

We found ChordPrism's grid-like UI to be both intuitively organized and user-friendly, providing a seamless experience for first-time users, seen in figure 2.12 below. Drawing inspiration from this plugin, we decided to adopt a similar layout for the project, positioning parameters at the bottom and chord display at the top. We initially considered developing a keyboard interface similar to ChordPrism, but ultimately opted to instead prioritize the chord display due to time constraints during development.

Figure 2.12: *ChordPrism's User Interface*

2.3.4.c ValhallaSupermassive

Another source of inspiration for the plugin's design was ValhallaSupermassive. This plugin is renowned for its many options for reverb and delay (*Valhalla Supermassive*, n.d.). Its design incorporates the use of dial sliders, which allow the user to adjust the parameters by small units, increasing the amount of options for the user. The dial-slider design made efficient use of the plugin's size window, and was user-friendly as the dials were easy to use and adjust. Figure 2.13 shows the full UI for ValhallaSupermassive.

For our development process, we drew inspiration from this plugin. The effectiveness of the dial-sliders played a great role in our decision to incorporate a similar organizational structure for their UI. It allowed for a visually appealing interface that includes multiple parameters that can be adjusted at any time. This plugin also includes style presets which automatically adjust each of the sliders to a designated position. We ultimately included this feature in their final deliverable.

Figure 2.13: *ValhallaSupermassive's User Interface*

2.4 Implementation Frameworks

Our project is geared towards people making music within DAWs. To offer a seamless experience across multiple platforms, an audio plugin (VST3 format) is the best solution. Audio plugins meld effortlessly into DAW ecosystems, enhancing usability. Conversely, a standalone or web app would require users to upload MIDI melodies, adjust chord preferences, and then download the resulting MIDI chords—a process that's not just tedious but also interrupts the natural flow of composing.

2.4.1 Max

Max is a graphical programming environment primarily for real-time audio processing and interactive multimedia applications. With its visual-based approach to programming, Max provides users with a collection of objects that are connected together in a patcher environment (see Figure 2.14 below).

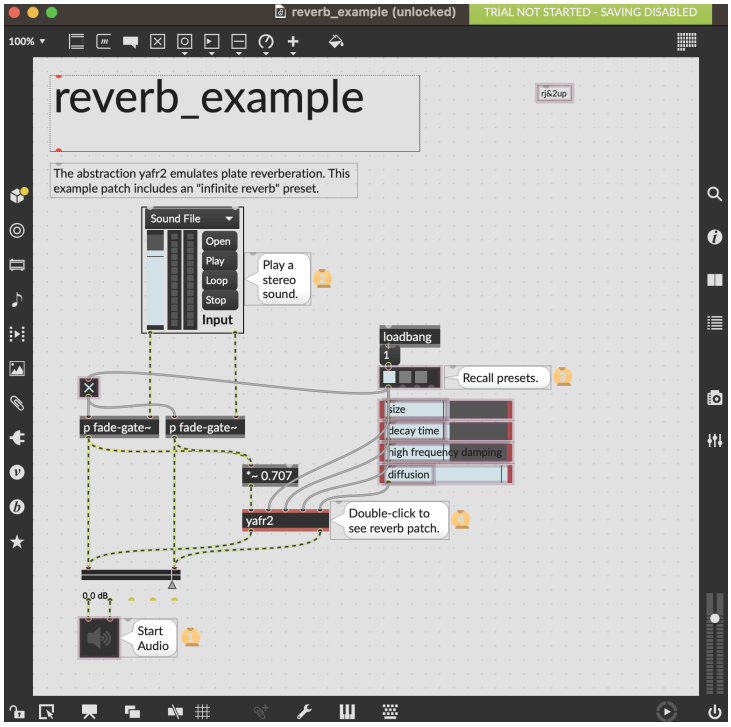


Figure 2.14: *Screenshot from Max*

Programs made with Max (called patches) integrate seamlessly into the popular DAW Ableton Live using Max for Live. It allows Max patches to be used as either a Max Instrument, Max MIDI Effect, or Max Audio Effect which can be loaded in Ableton Live and mimic the functionality of a VST. There is a large library of over 6000 Max for Live devices created (Cycling 74, 2023). There is also RNBO for Max which makes it possible to export a Max patch as a VST3 format plugin (Cycling 74, 2022).

2.4.2 JUCE

VSTs are programmed using C++, and "JUCE is the most widely used framework for audio application and plug-in development. It is an open source C++ codebase that can be used to create standalone software on Windows, macOS, Linux, iOS and Android, as well as VST, VST3, AU, AUv3, AAX and LV2 plug-ins." (JUCE, 2024) It is used by Google, Meta, Fender, THX, and many more (JUCE, 2024). At its core, JUCE is valued for its ability to allow developers to write code once and then compile it for different operating systems. This cross-platform capability is helpful for developers who want their applications and plugins to be accessible in different formats without needing to rewrite their code for each platform. Projucer

is the project configurator tool that ships with JUCE which allows developers to generate project files for different IDEs, such as Xcode on macOS and Visual Studio on Windows. This capability is particularly beneficial for teams with members working on different operating systems. We also considered Steinberg VST SDK, iPlug 2, HISE, Cabbage, and Faust. However, a main focus for our project is having the ability to drag a MIDI file into our VST, do chord generation and then be able to drag the output MIDI chords onto a track in the DAW. JUCE had all the features to make this functionality easy to implement.

2.5 Ethical Implications

There are many different types of ramifications within this industry to consider. With the rise in AI's popularity over the past few years, particularly in the arts sector, also comes a rise in skepticism and worry about the culture of the art form. The music industry is potentially entering a new era, for better or worse. This section explores the issues with AI's involvement in the music industry and offers insights into potential ethical considerations.

Before the various topics of AI are discussed, first some issues about the online marketing of VSTs must be inspected. Upon completion of assessing the current VST market, there are a number of VSTs that are being falsely marketed as AI or otherwise complex chord generators. A good example of this is Scorch, who advertises their chord generator, when in reality it is a collection of samples and presets (Scorch Plugin). It is normal to sell sample packs or MIDI packs, but the wrongdoing lies in the false marketing. Scorch and many other companies like them are trying to capitalize on trending AI by targeting inexperienced music producers looking for quick fixes to improve their craft. These companies are not being transparent about what their products are capable of, and leave the finer details behind a paywall of typically $100 or more. Considering that there are plenty of real AI tools that are either cheaper or free to use, it shows that these companies are relying on their customers' lack of knowledge. This emphasizes the importance of transparency about a product, and fair pricing to boot.

The largest ethical concern for AI in music and arts is that of plagiarism. Ideally, a metric of success for AI is how well it can imitate a style without copying it (Yin, 2021). Naturally, this line tends to get blurred, and the possibility for plagiarism, even if accidental, is everlooming. It

is tough to identify the plagiarism as well; deeply similar melodic and harmonic contours appear in different musical works all the time by complete coincidence as opposed to deliberate copying (Yin, 2021). Even when certain musical motifs are deliberately copied, it is usually meant as a clever nod to that artist, such as when a jazz musician quotes the lick of another artist in their solo, or when a sample is used on a hip hop track. However, when an excerpt is directly copied by an AI, there is no sort of intent or meaning behind it; it is simply a machine that is pulling from a source material as instructed. Researchers have found that some algorithms, especially ones using deep learning, have a recurring problem of directly copying lines from their training data in their outputs (Yin, 2021). This becomes particularly concerning when considering that there may be unknowing, unconsenting artists whose work are in this training data; it is nearly inevitable that this happens.

Luckily, this is a less prominent issue when it comes to harmonic generation as opposed to melodic. Many songs share the same chord progressions, and creating a song with a similar progression to another is not as suspect as copying another song's melody. In fact, the act of writing a new melody over an existing progression is called a contrafact; contrafacts are very common. That being said, measures should still be taken to mitigate it. For starters, our project will include many different custom parameters adjusted by the user, which makes the output not solely determined by the training data. Philosopher and researcher Octavio Kulesz (2018) argues that this human intervention is important not just for plagiarism concerns but also so the music is not "void of intent" (p. 5). He claims that "the most effective formula is collaboration between human being and machine" (Kulesz, 2018, p. 5). When it comes to the training data itself, it is constricting to rely only on works that are in the public domain, but any artists whose work is used to influence generation should be credited.

On top of this, artists should not deceive their audiences, and if their works are generated with AI, the audience should be informed of what they're listening to (Norberg, 2023). This principle of transparency is not only ethical but also enhances the integrity of the creative industries. By openly acknowledging the use of AI in the production process, artists can foster a culture of honesty and trust between them and their audience. Such disclosure need not detract from the artistic value or the talent involved in creating the work. Instead, it can lead to a deeper appreciation of the innovative methods and collaboration between human creativity and

technological advancement. Moreover, this approach encourages an informed discourse about the evolving landscape of art and music, allowing consumers to make conscious choices about the content they engage with. In doing so, the relationship between artists and their audiences becomes more authentic, supporting a more sustainable and ethically responsible art world.

Another issue with AI's involvement in music is how copyrighting will be handled. According to lawyer Albin Gelhaar, one should not claim copyright at all on a solely AI-generated work, since they had no part in that art's creation (Norberg, 2023). However, with significant human intervention, if the work is considered transformed enough, then they can (Norberg, 2023). This coincides with how sampling currently works in the music industry; a sample must be used transformatively enough in order for the artist to claim the song as their own. The art of sampling has its own set of ongoing and still debated ethical issues, but the rules laid down by sampling etiquette could provide us with a potential road map to navigate some of these AI generation issues.

Not only is human intervention pertinent to plagiarism prevention and copyrighting, but also to musical culture. Kulesz (2018) points out that society should not be worried about a stereotypical AI takeover, but rather a takeover by the large companies powering it. These large companies have the power to push out unprecedented amounts of AI-generated art, and with increasing realism. If assigning copyright to an AI itself was allowed, then these companies could copyright everything they put out and quickly monopolize the entire industry (Kulesz, 2018). This could then lead to what he coins as the "perfect bubble", where people become complacent with all forms of art- and culture by extension- being fabricated with no human hand, thus killing the meaning of culture entirely (Kulesz, 2018, p. 6). As long as human collaboration is at the forefront, hopefully this issue will never come to fruition.

3.0 Methodology

The methodology section outlines our decisions for developing a VST plugin that generates chord progressions from input melodies, leveraging a Hidden Markov Model trained on a robust music database. Our process is divided into four main phases:

(1) Back-End, where we delve into the creation of the generative model using Python and discuss the data preparation, and the rationale behind choosing a probabilistic model over deep learning alternatives for enhanced customizability, and evaluation.

(2) Front End, detailing our decision to utilize the JUCE C++ framework for the user interface, our considerations for plugin format, how we designed UI parameters, and how we integrated the front and back end using a REST API.

(4) Evaluation Metrics, where we describe the quantitative evaluation of the generated chord progressions based on traditional music theory conventions.

(3) User Study, in which we describe the setup and execution of our user study for evaluating the usability of our software and the effectiveness of the generated chord progressions.

3.1 Back End

This section delves into the architectural design of our project's backend, developed with a focus on employing a Hidden Markov Model in Python for chord generation. We explore the backend's structure, detailing the critical decisions made during its development, including the choice of programming language, the model's framework, and the integration techniques used to connect it with the front end. Special attention is given to the model's data handling, from sourcing and preprocessing to its implementation for generating musical chords that accompany given melodies.

3.1.1 Data

The first step to training any machine learning model is to identify, clean and transform data. In order to train a model which takes melody notes as input and generates chords as output, we must find examples of such associations in real-world music. A lead sheet, which is a popular format of sheet music amongst jazz musicians, is a reduced form of traditional music notation which only contains a notated melody and chord symbols (*Why Lead Sheets? | Berklee,* n.d.). Since this format fits the model's needs perfectly, we decided to use a dataset which contains some digital representation of lead sheets. While other formats of sheet music could also be viable, like full orchestral scores, they would require more data transformation and cleaning to effectively reduce them down to a melody and chords.

The preferred format for storing digital sheet music data is MusicXML, which is a file specifically designed for music notation (McKinney, n.d.). As a result, there is a wide variety of datasets of MusicXML files that are publically available, including everything from early renaissance music to modern pop. Thanks to the prevalence of this data, a combination of two publicly available musicXML datasets successfully trained the model for this project.

By examining past approaches to developing similar models, we learned that the probabilistic and deep learning approaches are the most popular. While both models are effective in their ability to generate chords from a melody, deep learning models lack the capability of direct manipulation, which proves to be a challenge in creating software features which allow the user to customize the chord generation through a series of parameters.

Deep learning models, by some metrics, have been able to show more promising results when compared to their probabilistic counterparts. However, probabilistic models, such as the Hidden Markov Model, oftentimes provide the developer with direct, intuitive access to the inner-workings of the model, allowing for a greater sense of customization. In comparing the two approaches, a clear disadvantage can be seen with deep learning approaches, if direct customization is required. This is due to the "black box" phenomena that occurs, which is the idea that developers are unaware of exactly how a model which has been trained using deep learning actually makes decisions, which prevents them from being able to backtrace steps that may have occurred which led to problematic generations and to make changes which will

prevent such problems in the future. This also reduces the direct customization that a developer can have on the model, as the only way to modify a deep learning model is to change the training data or how the training data is interpreted through training parameters. Because of this, this project uses a HMM, which is described in Section 2.3.2.1. The decisions that are produced by an HMM can be traced down to a deterministic mathematical function which considers values from three elements- the emission probability matrix, the transition probability matrix, and the initial state distribution array. Each of these elements are simply collections of probabilities, which can be modified directly by simply changing the value of those probabilities. For instance, suppose one of the software features aims to reduce the occurrence of a particular chord in the output. In this case, the developer could directly and artificially modify the corresponding values in the matrices to achieve the desired outcome. This direct approach when compared to the modification of the training process which would be required for a deep learning model is more efficient and intuitive.

3.1.2 Using Python

For developing the Hidden Markov Model which serves as the backend of the software, we selected Python over C++. Python's abundant resources for music-related data processing drove this decision, as it’s a crucial aspect of our project. Many existing projects in the music domain have successfully used Python, which provided many examples which guided the creation of our model. While C++ offers performance advantages, the amount of existing music utility libraries and example projects in Python made it the optimal choice for our project.

3.2 Front End

This section outlines the development of our software's front end, structured into four main parts: the choice of plugin format, the rationale behind selecting the JUCE C++ framework, our approach to creating UI parameters, and the REST API used to integrate the front end and back end.

3.2.1 Audio Plugin Format

Selecting an appropriate format for the front-end design is crucial, as it serves as the primary bridge between the user and the model. We carefully considered various software

formats before deciding to create an audio plugin (VST3/AU) for its seamless integration into the Digital Audio Workstation (DAW) environment, which enables musicians to use the software directly within their preferred DAWs. We explored building a web-based or standalone application, but these alternatives were not ideal as they required users to switch between different applications, potentially interrupting their creative flow. An audio plugin would better ensure a smooth and intuitive experience for the musician during the composition process.

3.2.2 JUCE C++ Framework

After thorough experimentation with various frameworks, we selected the JUCE C++ Framework over alternatives like Cycling 74's Max and iPlug2. Max presented challenges in exporting plugins for multiple DAWs. Its add-on, Max for Live, is only compatible with Ableton Live which would limit the plugin to one DAW. JUCE's large support network also influenced our decision-making process. The JUCE forum has plenty of examples for reference along with active JUCE developers who can help answer questions. There is also a great abundance of YouTube resources for JUCE compared to iPlug2, which fell short in documentation and community resources.

3.2.3 UI Parameters

When creating the UI for our software, we carefully considered which parameters from the model backend should be made available for user interaction. We worked to make sure these parameters were intuitive and explored strategies to combine various detailed parameters into singular, more abstract controls that the user would find simpler to engage with. This approach aimed to present users with the ability to influence the software's output meaningfully while shielding them from the complexity of the underlying Hidden Markov Model (HMM) backend. By abstracting complex settings into user-friendly elements, we created an interface that prioritizes ease of use without sacrificing the depth of control.

To determine which parameters were the most important to use, we created five user personas, each with different names, careers, levels of experience and musical goals. Table 3.3 lists each of the personas that we developed. For each of these personas, we wrote two stories

that describe what each of the personas would like to do with the software. Appendix B includes the full list of user stories associated with each of these personas.

Table 3.3: *User Personas*

Name	**Role**	**Experience**	**Reason to use Product**
Jake	Music hobbyist	Comfortable with DAWs and VSTs	Generate chords quickly for personal use
Amy	Music teacher	Minimal DAW experience	Wants to teach students how to generate chords
John	Professional film score composer	Beginner DAW user	Looking to expand his toolset
Sarah	Audio engineer	Years of DAW experience and sound design	Wants to generate chords quickly on the job
Raj	DJ and electronic music producer	Very comfortable with music technology	Wants to create remixes

Amy and John are both beginner DAW users, so this increased the need for buttons that were clearly labeled with their purpose to be intuitive for new users. We also felt it was important to include advanced options, so that experienced DAW users would have a reason to

use the software. Ensuring the balance of easy accessibility and complexity was something we considered during the design process.

Our approach to Chord Craft's UI focused on minimalist design to ensure a streamlined and intuitive user experience. The selection of each UI element is based on the complexity of each parameter: dropdown menus were implemented for presenting multiple choices in a compact format, knobs were selected for parameters with more than three options, vertical sliders were used for parameters with only three options, and buttons were designated for binary choices. Additionally, we developed custom UI components to streamline the process of dragging and dropping files for input and output. To enhance usability, we strategically organized UI elements from left to right, mirroring conventional reading patterns to support an intuitive workflow.

3.2.4 REST API Integration

For integrating the JUCE C++ frontend with the Python HMM backend, we implemented a REST API using a Python server instead of directly embedding the Python model into the audio plugin. This strategy emerged from an assessment of the challenges in doing direct Python script execution within the audio plugin's C++ environment. The REST API provides an efficient conduit for communication between the frontend and backend, establishing a dynamic framework that supports ongoing development and future enhancements. This setup ensures any updates or modifications to the backend can be implemented without requiring alterations in the frontend. The flexibility of the REST API solidified its position as our preferred choice for the project.

3.3 Evaluation Metrics

Although musical harmony preference is subjective, we created a set of quantitative measurements of the model's output to compare to tendencies found in the dataset. The following metrics serve as generic indicators of how well the model is adhering to musical trends. These metrics were also carefully selected to evaluate all of the different types of data being processed by the model.

3.3.1 Diatonicism

All major and minor keys have unique sequences of notes that belong to them. These notes comprise the diatonic scale of the key. The diatonicism metric indicates the percentage of chord tones in a given collection of chords that belong to the given key. This metric is useful because it gauges the overall adherence of the chord progression to the key, without having to rely on the presence of the tonic chord.

This metric was also important to include considering how common tonal deviations are in music. An example of such is the usage of borrowed chords. A chord in a given progression is considered 'borrowed' when it originates from a different diatonic key than the song's key (Nettles, 128). The most commonly borrowed chords typically come from the parallel minor, such as using an F minor chord, which is diatonic to C minor, in a C major progression. The inverse, borrowing from the parallel major in a minor progression, is also typical. The process of using such borrowed chords is referred to as modal interchange (Nettles, 128).

Another frequently used harmonic technique which lowers the diatonicism percentage is the usage of modulations. A modulation occurs when the key of a song temporarily changes (Nettles, 146). This is different from modal interchange; in modal interchange, the original tonic reference remains, whereas in a modulation, the lister's focus is redirected to a secondary reference (Nettles, 146). In lead sheet notation, accidentals are commonly used as opposed to changing the key signature (Nettles, 147), which is why these modulated chords are still considered in relation to the original diatonic center instead of temporarily using the modulated key as a new reference.

Overall, the diatonicism will give a rough indication of key adherence, and how often the harmonies leave that tone center to explore other facets. A relatively high score suggests that the progression is very rigid and rarely strays from the seven diatonic chords, and a low score indicates that the progression isn't very tied down to one diatonic key center, for better or worse.

3.3.2 Note Fitting

The note fitting metric is the percentage of melody notes in an excerpt that are not an avoid note under their given chord. Melody notes can be classified into three categories: chord tones,

tensions, and avoid notes (Nettles, 17). A chord tone is a note that is found within the given chord, a tension is a note outside of the chord that can provide additional color, and an avoid note is a note that is generally dissonant with the given chord (Nettles, 17). These avoid notes are considered dissonant since they typically destabilize the chord (Nettles, 177). For example, playing the 4th scale degree over the major tonic sounds a suspension and resolution at the same time, causing destabilization (Nettles, 177). Avoid notes are typically used sparingly as passing tones, as opposed to being the focal tone of a melody.

It is important to compare the occurrence of avoid notes in the model versus the dataset in order to gauge how well the model follows this convention. While this classification is derived from jazz theory, jazz has provided the basic harmonic foundations for many forms of modern Western music to this day, so this convention is present in nearly any popular American genre, such as pop, hiphop, country, and so forth. With an art form such as music, there are always exceptions to rules, but this still provides a well-encompassing indicator of how well the melody notes blend in with their assigned chords.

Table 3.1: *C Major Avoid Notes*

Avoid notes for modes of the C major scale

Scale degree	Chord	Mode	Avoid note	Available tensions
1	C^{maj7}	Ionian	Fourth scale step, F	9, 13
2	Dm^{7}	Dorian	Sixth scale step, B	9, 11
3	Em^{7}	Phrygian	Second and sixth scale steps, F and C	11
4	F^{maj7}	Lydian	No avoid note	9, ♯11, 13
5	G^{7}	Mixolydian	Fourth scale step, C	9, 13
6	Am^{7}	Aeolian	Sixth scale step, F	9, 11
7	$B^{ø7}$	Locrian	Second scale step, C	11, ♭13

(Nettles)

Table 3.2: *Melodic Minor Avoid Notes*

Avoid notes for modes of the melodic minor scale

Scale degree	Chord	Mode	Avoid note	Available tensions
1	Cm^{maj7}	Ionian, ♭3	No avoid note	9, 11, 13
2	Dm^{7}	Dorian, ♭2	Second scale step, E♭	11, 13
3	$E♭+^{maj7}$	Lydian augmented	Sixth scale step, C	9, ♯11
4	F^{7}	Lydian, ♭7	No avoid note	9, ♯11, 13
5	G^{7}	Mixolydian, ♭6	Either the fifth or sixth scale step, D or E♭	9, 11
6	$A^{ø7}$	Locrian, ♯2	No avoid note	9, 11, ♭13
7	$B^{ø7}$	Super-Locrian	No avoid note	(usually played as altered dominant)

(Haerle)

3.3.3 Common Tones

A common tone is a note that appears consecutively from one chord to the next. This metric is measuring the frequency of common tones. For example, going from a C major triad to G major evaluates to ~33%, since the C chord shares ⅓ of its notes with the G chord.

The presence of common tones in harmony can provide all sorts of contextual information. The more common tones that two chords or keys share, the more related they are to each other (Woodruff, 84). The presence of common tones can be used as an indicator towards function. With extensions included, the III- and IV- chords, which also fall under the tonic, share the most common tones with the I. The subdominant chords, II- and IV, share less, and the dominant chords, V and VII-, share the least. Therefore, the frequency of common tones and the degree of harmonic function/motion share an inverse relationship. It is therefore important to compare this statistic across the model and dataset, to see how well the model is following trends of harmonic motion.

3.4 User Study

To assess the effectiveness of the software, we designed a user study to administer to members of the WPI community with an interest in music. The goal with the study was to see participants use the software in real time, as well as to give the participants the opportunity to

provide feedback about the software. To best achieve this goal, we decided on using a talk-aloud and a survey as the methods. Appendix C lists the full protocol of the methods.

The first part of the study, the talk-aloud, is a usability method that encourages the participant to verbalize their thoughts while moving through the user interface (Nielsen, 2012). Before the activity, we asked the participant a series of introductory questions to assess their prior experience with plugins, music theory, and composition. We structured a guided video tutorial of the software, as well as a series of objectives to complete. These objectives aimed to introduce the user to the different parameters, and to familiarize them with uploading a MIDI, generating chords, and downloading the result. To complete the talk-aloud, the participant used the VST on Ableton Live which was on a supplied computer. We captured screen and audio recordings of each participant for the purpose of interpreting data. The study consisted of the participant experimenting with three different melodies, composed prior to the study. The participant navigated through tasks such as adjusting parameters to create progressions of varying keys and complexities. The goal of this part of the study is to assess the usability of the software features, and understand how a first-time user interprets the instructions.

The different questions in the talk-aloud aimed to assess how concise the parameters were labeled without us explaining the purpose of each one. We wanted to design the questions in a way so that they increase in difficulty as the participant grows more comfortable with the software. We also wanted to give the participant an opportunity to write their own melody and generate chords for it.

Following this process was the concluding survey, where the participant had more opportunities to share their thoughts about the software. We aimed to gather information about software feedback and experience. The questions targeted the participant's opinions on specific parameters, as well as gauging interest in this software as a commercial platform.

At the conclusion of the study, we transcribed the recordings of both the participant's screen and their responses from the interview. The next section describes the results of both the talk-alouds and the semi-structured interviews.

The analysis of the collected data requires several key steps to ensure the collection of the most important information from each of the participants. The first step in the process was

transcription, which required us to use an automated transcription software of what the participant said, while double checking that the software produced accurate transcriptions. After transcription, the text will be coded into the following major categories: UI feedback, chord satisfaction, and general experience. Each of these categories may be divided into subcategories where deemed appropriate.

From the questions in the intro survey, we made graphs to visualize the quantitative responses from the participants, as well as the comparison of the different styles that each participant indicated listening to. We also made bar graphs for the rest of the questions to compare responses across all participants. We then split up each of the four tasks in the talk-aloud to further analyze each of the responses.

4.0 Results

This section presents an overview of our software's architecture, comprising two primary components: the front end, developed using the JUCE C++ framework, and the Python-based HMM chord generator serving as the backend. We detail the decisions made regarding the setup, structure, and parameters of the frontend, and provide insights into the requirements of the model and the algorithm underneath the backend, detailing the essential functionality of the plugin.

4.1 Back End: Python HMM

The first part of the software is the Python HMM. This section of the paper provides an overview of the structure of the back end as well as details regarding the generative model.

4.1.1 Back End Structure Overview

The generative model, described in the next section, is designed as a RESTful API (*What is RESTful API?,* n.d.) using Flask, a Python library. The RESTful API allows clients to send parameter values and melody information in the form of a MIDI file to a server as a GET request, which then returns the generated chord progression back to the client as a POST request.

4.1.2 Model Requirements

The primary goal of the model is to generate a chord progression that fits with a melody. Additionally, there should be several parameters that can be adjusted by the user to allow for customization of various musical elements of the generation.

To train such a model, the data format must be defined. In this case, since we are looking to generate chords from input melody notes, the training data can be identified as a series of note-chord sequences from existing songs. This data is highly sequential, since the order in which the data occurs is significant, as the order of notes and chords in a song is an important context to capture the essence of the song.

The dataset used was a combination of two datasets (*shiehn/chord-melody-dataset*, 2019) (*00sapo/OpenEWLD*, 2020), consisting of 815 popular songs in the public domain. Generally,

the songs in the dataset come from the Great American Songbook, late 20th century American pop and Christian music. Each song is in MusicXML format, which is an open-source standardized format for music notation, often used by musical notation softwares (MusicXML, 2021).” MusicXML files contain information about the song, including the sequence of notes, rests and chord objects which make up the song, time signatures, and much more. The primary advantage to using MusicXML versus Music Instrument Digital Interface (MIDI), another common digital music format oriented towards live performance, is that MusicXML files contain specific information about the length of each note and rest and have support for distinguishing the notes which come from voiced chord symbols versus main melody notes. In this dataset, songs are in lead sheet format, which is an “...abbreviated form of music notation…[consisting] of just the melody or lead line and chord symbols (Feist, 2018).”

Music21, a Python toolkit for computer aided musicology (Music21, n.d), is a Python library developed by the School of Humanities, Arts and Social Sciences at the Massachusetts Institute of Technology which can be used to parse and extract data from MusicXML files. Using the Music21 parser, the model is able to extract all of the necessary training information from the files - namely the chord, note and rest info.

4.1.3 Hidden Markov Model

The software uses a Hidden Markov Model (HMM), as described in section ___. HMM’s are well-equipped to analyze sequential data (hmmlearn, n.d.). The model was created using HMMLearn, a scikit-learn inspired Python library (*hmmlearn—Hmmlearn 0.3.2.post3+g97c2fc5 documentation*). The data was transformed using various NumPy and Music21 functions in Python. NumPy is a Python library which adds support for easily modifying multi-dimensional arrays.

The notes were identified as the observable states and the chords corresponding to those notes were identified as the hidden or unobservable states. This enables the model to predict sequences of hidden states which are most likely to occur, given a series of observable states - essentially, predict a series of chords from a series of melody notes. An HMM consists of several components - a set of observable states, a vocabulary of hidden states, the transition probability matrix representing the probability of moving from state to state (ie. chord to chord), the emission probability matrix representing the likelihood of a hidden state (a chord) occurring

based on a particular observable state (a note), as well as the initial state distribution probabilities.

The vocabulary that makes up all of the possible hidden states consists of each of the following chord types with each of the 12 pitch classes as the root (Table 4.1). To avoid bias with different keys, each chord is recorded relative to its pitch class within the key of the song. This is most effective when each song remains in the same key, although in theory, key changes could be learned by the model within songs. To populate the transition probability matrix, where the x axis refers to the starting chord and the y axis refers to the chord immediately following the starting chord, each song is iterated through and a count of the frequency that each chord transition is recorded in the matrix. After all the songs in the dataset have been iterated through, the matrix values are normalized.

Table 4.1: *A list of all Supported Chord Types and Common Notations*

Chord Type	Annotation
major	N/A
major-sixth	m6
major-seventh	M7
major-ninth	M9
major-6/9	6/9
minor	m
minor-sixth	m6
minor-seventh	m7

minor-ninth	m9
augmented	+
augmented-seventh	7+
diminished	o
diminished-seventh	o7
half-diminished-seventh	m7(b5)
major-minor-seventh	m(M7)
dominant-seventh	7
dominant-ninth	9
dominant-11th	11
dominant-13th	13
power	5
suspended-fourth	sus4

The set of observable states takes the form of a set of vectors representing all of the possible note combinations that could occur. Each state represents a set of notes that occur during the duration of a chord. Since there are 12 pitch classes, each state can be represented as a binary vector where each index corresponds to a pitch class and a value of 1 represents the presence of that note in the state. To populate the emission probability matrix, a similar process to the transition matrix is followed. Additionally, to populate the initial state distribution, the frequency of each chord occurring in the dataset as an initial chord is recorded at this step. Each state can be converted to a base 10 number by converting the vector to a binary value, then translating that into base 10. This number corresponds to the index in the matrix in which that state is represented. The format of the data is such that a chord object appears before all of the notes that

appear during the duration of the chord - therefore, each object can be iterated through and each occurrence of a state that occurs during a chord can be recorded then normalized in the matrix.

To generate a prediction, the MIDI input is transformed into binary vectors in the same form as the observable states. Then, this can be fed directly into the model which produces an output of hidden states, which are chords.

4.1.4 Voice Leading Algorithm

The voice leading algorithm used was developed by Dmitri Tymoczko as voiceleading_utilities (Software | Dmitri Tymoczko, n.d) The algorithm takes in specific pitches for the origin chord, and pitch classes (notes with no assigned octave) for the destination chord, and outputs the destination chord with specific pitches, voiced in a way which minimizes the distance traveled in pitch of the voices from the origin to the destination. The ChordCraft voice leading function takes a basic root position voicing of the first chord in the progression and the pitch classes for the next as input. It takes the outputted specific pitches from the previous destination, and makes those the new origin, and this process is repeated until the last chord in the progression is reached. Finally, the entire process is run a second time, starting with the last chord of the progression in order to generate a voicing in which the first chord is not in root position.

To accomplish this, a bijective function is defined as the distance between a set of pitches representing the first chord and a set of pitch classes representing the second chord. In order to find the chord voicing for the second chord which minimizes the distance between it and the first chord, each possible rotation or inversion of notes for the second chord is compared against the first chord. The distance between two chords is defined as the absolute differences between corresponding notes.

4.1.5 Parameters Functionality

Each parameter is designed to modify some aspect of the chord generation based on the user’s preference. The purpose of these parameters is to give the users more flexibility and customization. The functionality of each parameter is described below.

4.1.5.a Key and Tonality

The key and tonality parameters allow the user to select the key of the melody. This is essential information for the model, as it determines important context for how the input melody notes should be interpreted and the key in which the resulting chord progression output should be in.

4.1.5.b Alternative

The alternative parameter allows the user to select between a series of alternative outputs for the same input, allowing for more creative freedom and flexibility when using this program. Specifically, the user may select from the 1st, 2nd, and 3rd most likely chord transition outcomes, according to the Hidden Markov Model.

4.1.5.c Complexity

The complexity parameter affects the tendency of the algorithm to choose simpler chords, such as triads, versus more complex chords with more extensions. This is done by applying a scaling factor to the initial state, probability matrix and transition matrix, raising or lowering the probabilities of more complex chords in relativity to triadic chords. The user is offered an input scale from 1-10, 1 being the simplest and 10 being the most complex. At a value of 5, the model is completely unweighted. This was implemented to help better pursue the style of music that the user intends. Different styles of music tend to use varying levels of chord complexity; for example, classic rock music uses simpler chords on average when compared to jazz, a genre that is characterized by its harmonic complexity.

4.1.5.d Chords Per Bar

The chords per bar parameter determines how many chords will be generated for a given measure. The user can set it to 1, 2, or 0.5, which would be one chord for every two measures. The information given by the user changes how many beats the algorithm waits before cutting off a state and forming a new one. This parameter was included to provide more flexibility with the flow of chord changes, and those three different options are commonly used when composing music.

4.1.5.e Voicing

The voicing parameter changes the voicings of the exported MIDI chords. The voice leading itself is the same; only the octave placement of individual voices are affected. The user is

given three options to pick from: open, normal, and closed. When open is selected, the voices will be voicing out across many octaves, normal is more reigned in, and closed nearly always confines the voices to a single octave range. The voicing parameter allows the user to control the feel of the voicing without overwhelming them with too many options. In a similar fashion to complexity, different styles of music typically call for different voicings.

4.1.5.f Bass

The bass parameter allows the user to toggle whether or not the output contains a simple bass line, which is simply the root of the chord as the lowest note. This can generally result in a more complete sound which is rounded out by the bass voice. However, some users may prefer to write their own.

4.2 Model Evaluation

When properly assessing the model's ability to recreate harmonic trends found in human songwriting, it is important to know how the evaluation metrics are being calculated. It is equally important to be able to synthesize conclusions from the data's trends.

For the purposes of this project, the dataset calculations are made from the dataset in its entirety. For metrics derived from the model's output, a subset of 20 random songs from the dataset were used. The generator was prompted to use default mid-values for all parameters, and to take the first most probable generation.

When evaluating the model output, an intermediary output format was used. Before the final chord progression is transformed into MIDI, the chords are stored as a list of MIDI pitches. All of the quantitative evaluations and calculations are done using this list of MIDI pitches.

4.2.1 Diatonicism

When calculating this metric from the dataset, the percentages are being derived from the initial state distribution, which contains the probability of any individual chord occurring under the given key. All songs in the major dataset have been transposed from their original key to the key of C for consistency's sake. For the minor dataset, all songs are transposed to A minor, the relative minor to C. Each chord is broken up into its chord tones, which are the individual pitch

classes that comprise it. These chord tones are held against the notes that belong to the diatonic scale. After getting the percentage of the chord tones that also belong to the diatonic scale, it is then scaled by the probability of that chord actually occurring, then all of these percentages from each chord are added together.

When obtaining the diatonicism metric from a given model generation, it uses the outputted chords and the inputted key. The diatonic scale is derived from the key, and then the chords are broken down into their tones. The mean of the percentage of notes from each chord that align with the diatonic scale is calculated.

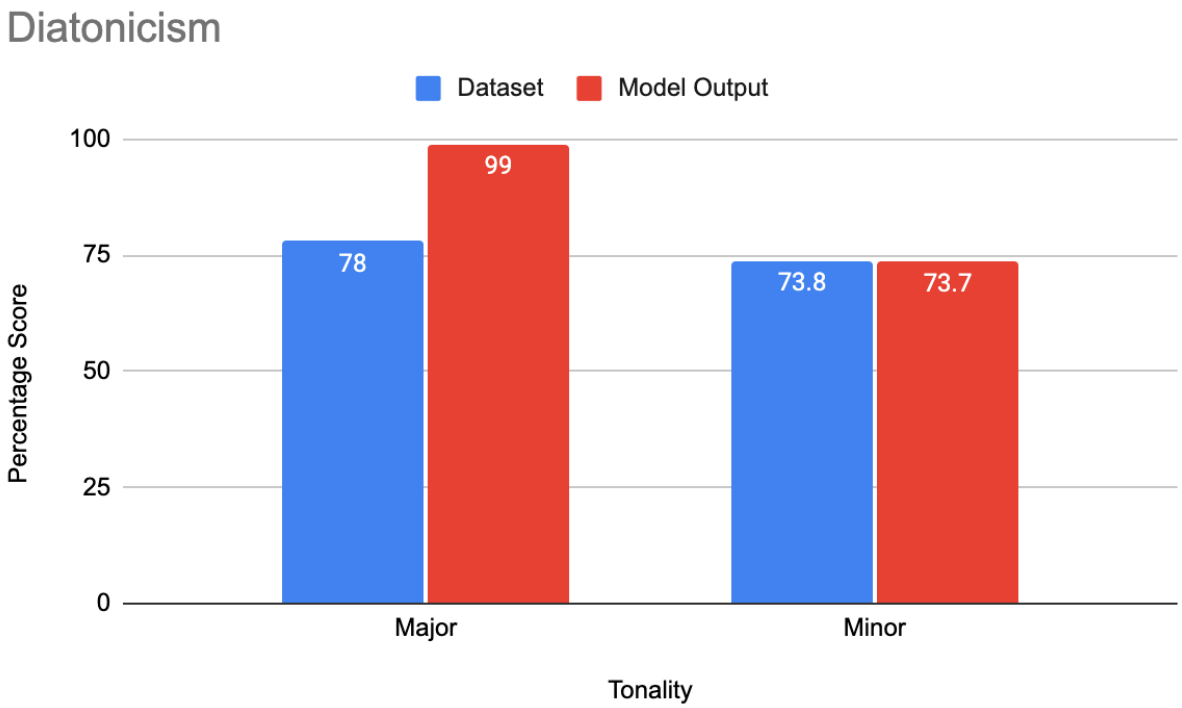


Figure 4.1: *Graph of Diatonicism*

As observed by the measured data, the minor dataset and model are nearly identical. There's a larger difference in the major, indicating that the model adheres to the diatonic scale of the key very strictly, at almost 100%. Meanwhile, human-written compositions are much more likely to deviate from that. The model here is overperforming, which is not preferred, but this is far from being detrimental to practical function; underperforming would be greater of an issue.

However, it is worthy to note that because of this, the model may not be taking as many creative ‘risks’ as a human composer would.

4.2.2 Note Fitting

When calculating this metric from the dataset, the percentages are being derived from the emission probability matrix, which holds the probabilities of any grouping of melody pitches occurring alongside any given chord. Each melody grouping is represented by a 12 digit long binary number, each digit representing one of the 12 pitch classes, with a 1 meaning that pitch is present. First, the percentage of notes in this binary vector that are classified as avoid notes under the associated chord is calculated. Then, this percentage is scaled by the probability of that vector occurring. All of these resulting percentages from every vector-chord pairing are added together. Finally, the reciprocal of this percentage is taken in order to get the percentage of melody notes that are not considered avoid notes, giving a note fitting metric.

To obtain the metric from a given model generation, it uses the outputted chords alongside the inputted melody notes. The percentage of melody notes that are avoid tones under their assigned chord is calculated for each chord, then all of these are added and the reciprocal is taken.

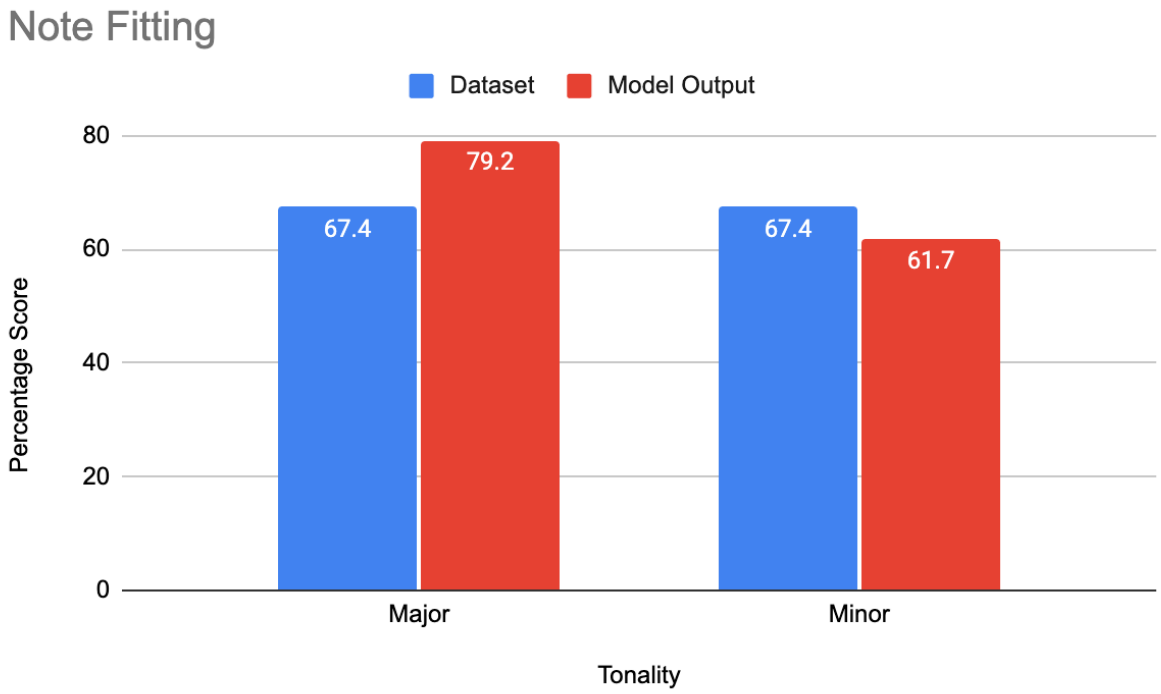


Figure 4.2: *Graph of Note Fitting*

For note fitting, there's a slight overperformance for major and underperformance for minor, but these differences are somewhat negligible when it comes to affecting practical function. The overperformance for the major model shares the same sentiment as for diatonicism; the model may be more risk-averse, for better or worse. The percentage difference for the minor model is realistically too small to make an observable difference.

4.2.3 Common Tones

When calculating the common tones metric from the dataset, the percentages are being derived from the transition probability matrix. This matrix contains all of the probabilities of one chord transitioning to another chord. The percentage of pitches shared from one chord to the next are calculated, then scaled by the probability of that chord-to-chord transition occurring. All of these resulting percentages are then added to get an overall percentage of common tones throughout the whole dataset.

To obtain the metric from a given model generation, the output chords are needed. The individual percentage of common tones from each chord to the next in the progression is calculated, then the mean amongst all of them is calculated.

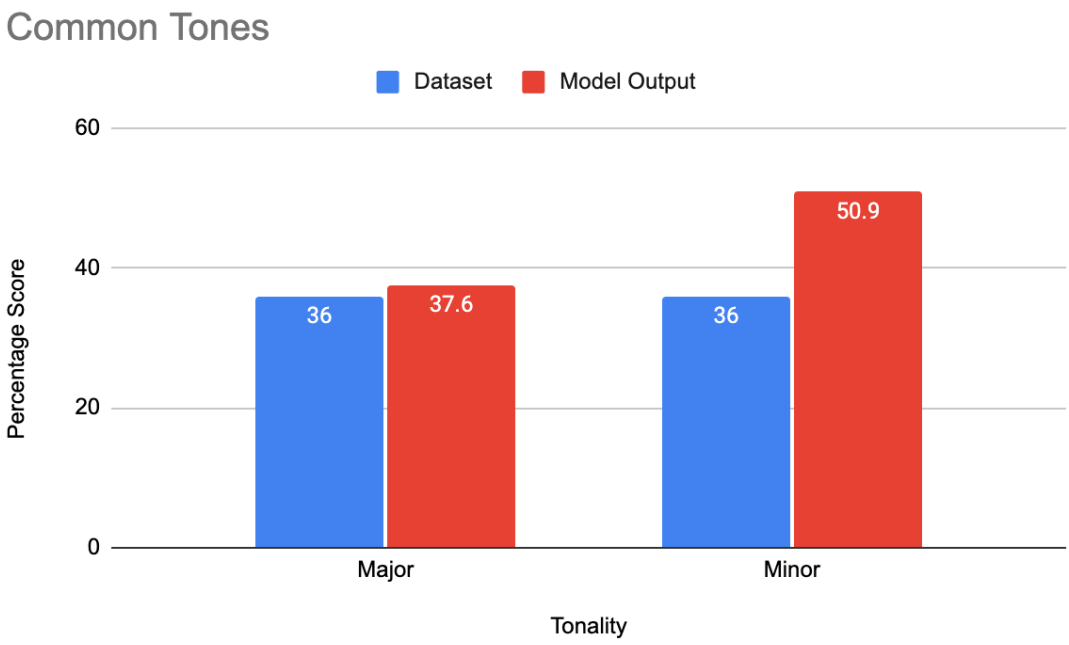


Figure 4.3: *Graph of Common Tones*

The minor model for common tones is notably overperforming here. This is most likely a result of repeated chords- every time a chord occurs twice in a row, the recorded common tone percentage for that instance is 100%. Anecdotally we have noted that the minor model is more likely to repeat chords in any given generation, which aligns with what is shown above.

4.3 Front End: JUCE Audio Plugin

The development process required us to become familiar with the JUCE C++ framework for the frontend design. We describe an overview of the setup process, highlighting the strategies used to ensure seamless compatibility across various operating systems. We then detail the protocol of integrating the parameters into JUCE, and highlight the development testing process.

4.3.1 JUCE Setup

For our development setup, we utilized the JUCE framework and its Projucer app for cross-platform development. We focused on refining the workflow within macOS and Xcode, while also ensuring compatibility with Windows via Visual Studio. This section details a JUCE and Projucer intro, working with macOS and Windows, Projucer vs CMake, and our automated build scheme.

4.3.1.a JUCE and Projucer

Our plugin is built upon the JUCE Basic Audio Plug-in project template. The framework comes with a project configuration tool called the Projucer which allows the project to be opened in different IDEs across multiple operating systems and have the application be built in various formats. In the Projucer screenshot below (Figure 4.4), we have our project configuration which allows our singular codebase to be compiled with either Xcode (macOS) or Visual Studio (Windows) and built as a VST3 or AU plugin.

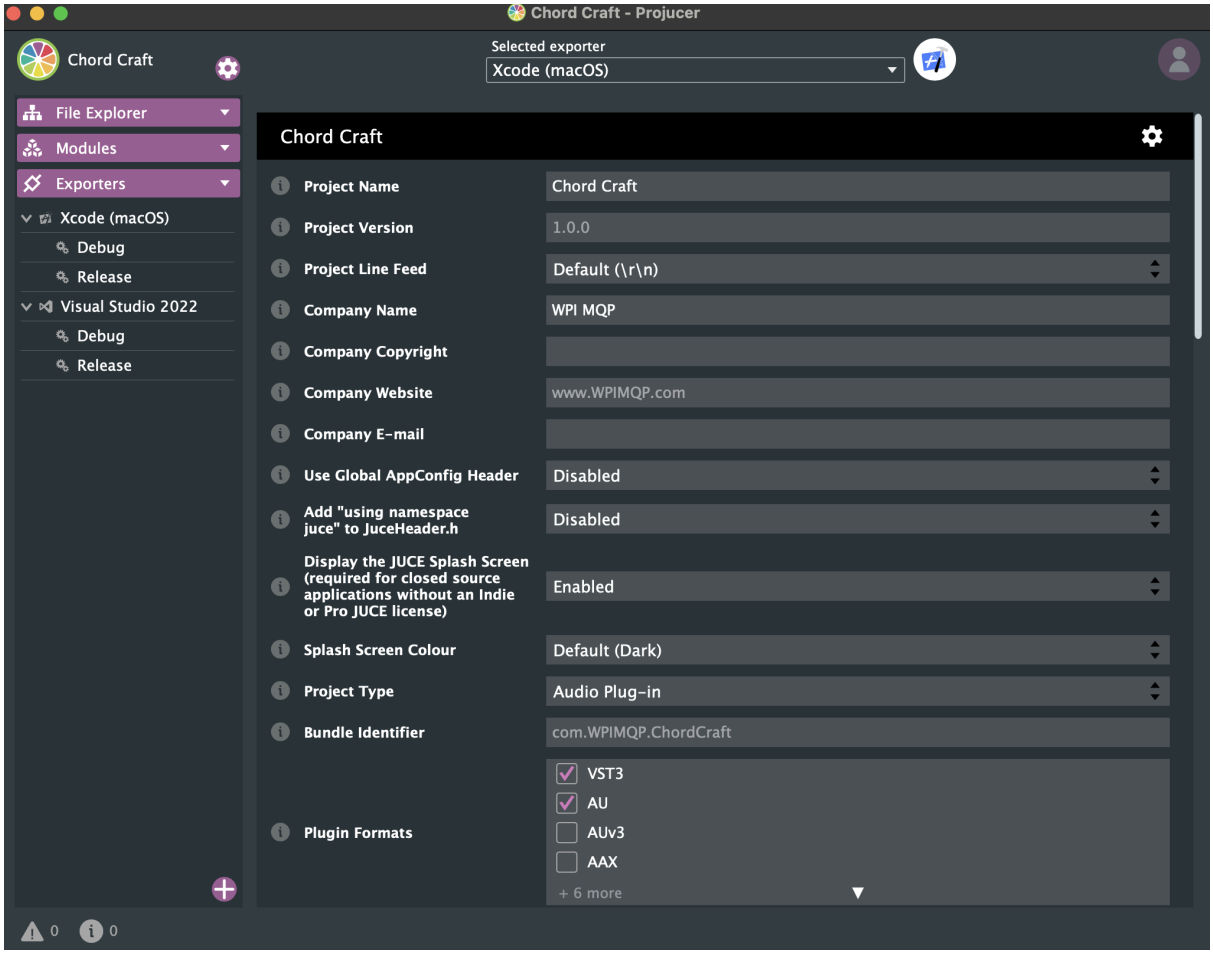


Figure 4.4: *Screenshot of Projucer Project Configurator Tool on MacOS*

4.3.1.b MacOS and Windows

Our plugin development mainly took place on macOS, utilizing Xcode for development and Ableton Live for testing. The versions used in development were: macOS Sonoma 14.1.1, JUCE v7.0.9 with C++17, Xcode 15.0.1, and Ableton Live 10. We tested the plugin across multiple DAWs and formats including as a VST3 in Ableton Live on Windows and macOS, VST3 in Reaper on macOS, and as an AU plugin in Garageband and Logic Pro on macOS.

4.3.1.c Projucer vs CMake

We also researched building projects with CMake as we initially wanted to use Visual Studio Code which we are all more familiar with (See Appendix A JUCE Cmake links). This way the build details for each OS would be specified in the CMake file. However there was less documentation on this build method and we faced many technical challenges while getting basic

examples running, so we decided to stick with the mainstream method of using the Projucer for developing our plugin.

4.3.1.d Automated Build Scheme

We first set up the development environment in Xcode with an automated build scheme that would launch Ableton Live upon successful build of the VST. Additionally, we crafted an Ableton Live project template which contained a track pre-loaded with the VST, ensuring the newly built VST was instantly available each time Xcode opened Ableton Live. This custom build scheme streamlined the development process by directly opening the VST within Ableton Live, eliminating the need to manually open Ableton and load the VST onto a track.

Additionally for UI development, we utilized Projucer's Standalone build format, enabling us to compile and run our software as a standalone app with Xcode, avoiding the need to launch it within Ableton Live. This method significantly accelerated our UI refinement process, allowing for quick iterations and immediate visual feedback.

4.3.2 Plugin Structure

The JUCE Basic Audio Plug-in project starts with PluginEditor and PluginProcessor files already generated. We created the MidiProcessor to keep all MIDI related operations separate. The core functionality of Chord Craft is encapsulated within three primary files:

1. PluginEditor
2. PluginProcessor
3. MidiProcessor

4.3.2.a PluginEditor.cpp

This file creates the user interface for the plugin. It is responsible for positioning and rendering all UI elements such as sliders, buttons, and other controls. It also manages user interactions like detecting when a file is dragged onto the UI and changing internal values when a UI element is adjusted by the user.

4.3.2.b PluginProcessor.cpp

This file handles all the audio processing. Our project only works with MIDI files, so our PluginProcessor file remained unaltered. This way when our plugin is inserted into an audio track in a DAW, it ensures that audio signals flow uninterrupted through the plugin without any processing.

4.3.2.c MidiProcessor.cpp

This custom file contains the bulk of Chord Craft's functionality loading the input MIDI file, communicating with the server to generate the output, and sending messages to update the GUI when certain MIDI processing is completed.

These three main files are supplemented by two additional utility files CustomLookAndFeel and Presets.

4.3.2.d CustomLookAndFeel.h

This file customizes the plugin's visuals such as font, color, and shape of different UI elements. This file is where we really finalized the plugin's appearance after all the elements were positioned within our PluginEditor.

4.3.2.e Presets.h

This file contains five preset configurations that users can choose from as a starting point for chord generations (refer to Table 4.2 for full list of preset styles). These presets do not change the Key Choice and Tonality settings because the user needs to set those to fit the melody's key. However, they do change five specific parameters: Alternative, Complexity, Chords Per Bar, Voicing, and Bass. The names of these presets suggest a general association with certain musical genres, reflecting typical chord patterns found within those styles, but it's important to understand that these connections are not guarantees of achieving a specific genre's hallmark sound. The presets are mainly a starting point for users to showcase how different parameters can change the chords generated.

Table 4.2: *Parameter Information*

Preset	Alternative	Complexity	Chords Per Bar	Voicing	Bass
Default	1	5	1	Normal	Off
Pop	2	1	1	Closed	On
Jazz	2	8	2	Normal	On
Hip Hop	3	4	0.5	Normal	Off
Chaos	3	10	2	Open	On

4.3.3 UI Iterations

The next objective of this project was to create the User Interface (UI) to correspond with the back end software. Once all the required functionality of the product was solidified, we designed multiple iterations of UI mockups to determine the arrangement of parameters for the final UI. We started by drawing sketches of how we envisioned the software to look (see Figure 4.5 below). We coupled this process with research about important parameters to include in music software, as well as observing previously-existing softwares. Each week, we engaged in thorough discussions to further refine the design (see Figure 4.6 for draft UI). This iterative process allowed us to incorporate new ideas while taking into account the nuanced preferences and expectations of users varying in musical expertise.

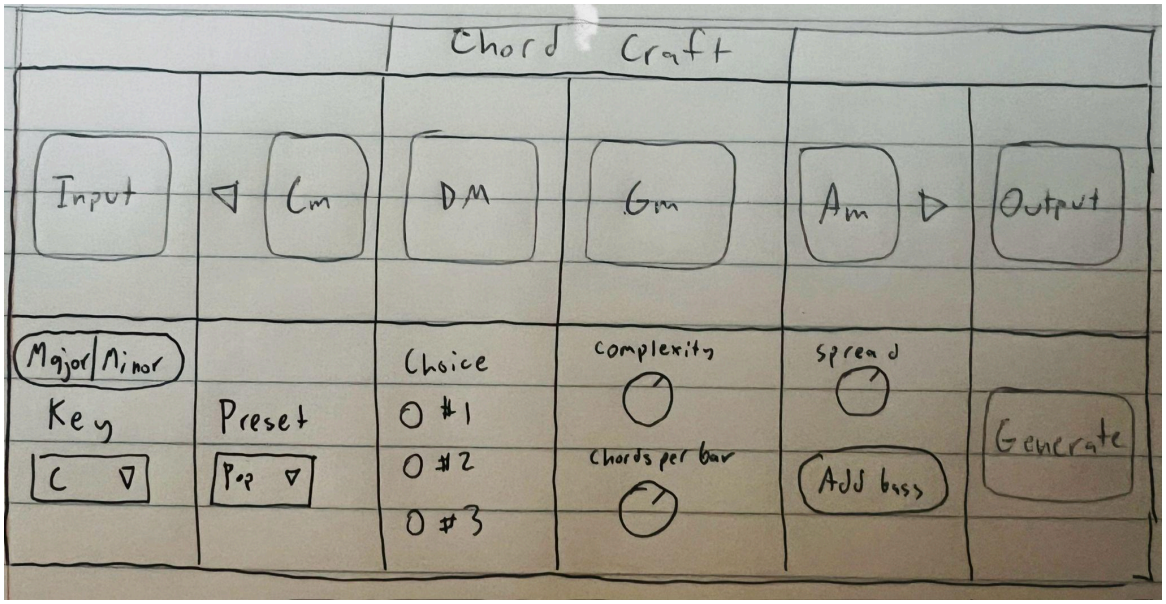


Figure 4.5: *Hand-Drawn UI Design*

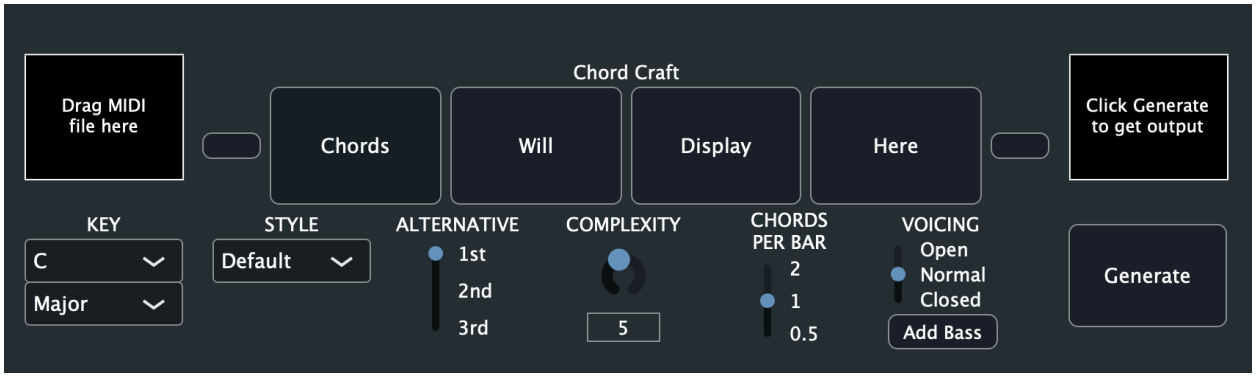


Figure 4.6: *Draft UI Iteration*

To provide the user with an intuitive experience, working left to right, we ordered UI elements in a specific order to influence the chord generation process (see final UI in Figure 4.7 below). Then different controls were used based on the number of options for each parameter. We used dropdown menus for the features with a lot of options, like Key and Style. Having dropdown menus allows the display to condense itself and ultimately conserve space. We chose to use vertical sliders for Alternative, Chords Per Bar, and Voicing as they have only three possible options. Then a knob was used for Complexity as it has ten possible options. The Bass parameter is a button toggle since it is a binary choice for the user. Then the chord display has two triangle buttons on either side to navigate to the previous or next set of chords if there are

more than four chords generated. Finally, we created custom UI elements for the input and output areas where external files can be dropped or dragged from.The variety in display options for the parameters helped to distinguish them from each other and made the program simple for the user to understand).

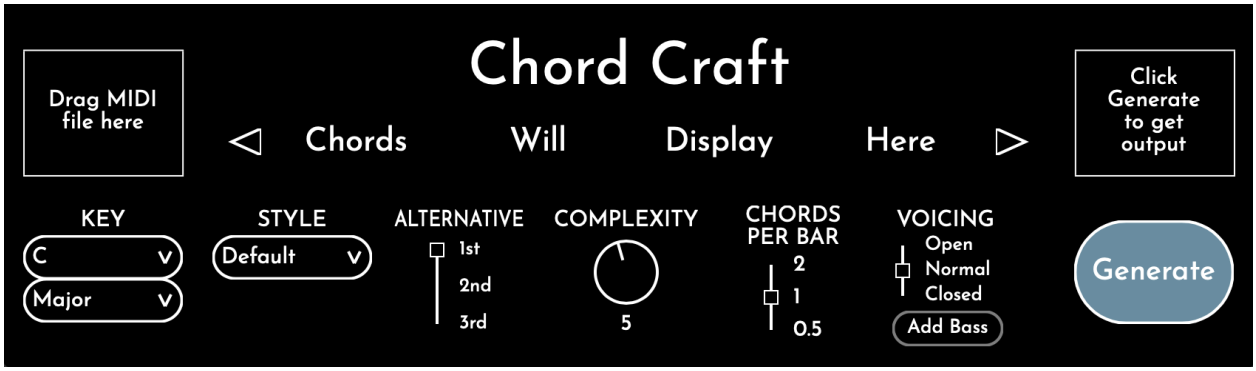


Figure 4.7: *Final UI*

4.4 Using ChordCraft

Within the Chord Craft plugin's user interface, once a user has prepared a MIDI file with a melody, the process unfolds as follows: Users initiate the process by dragging and dropping their MIDI file into the input area (see Figure 4.8 below).

Figure 4.8: *Dragging File into Input Area*

The PluginEditor first verifies the file's format and placement, ensuring it's a valid MIDI file (.mid or .midi) dropped within the input area, then the file gets passed to the MidiProcessor

which checks that the MIDI file contains only a single track. Upon successful validation, the PluginEditor updates the input area with a blue highlight and the file's name.

Next users adjust the key choice and tonality parameters to ensure the chords generated will match the correct key of the melody input given (see Figure 4.9 below).

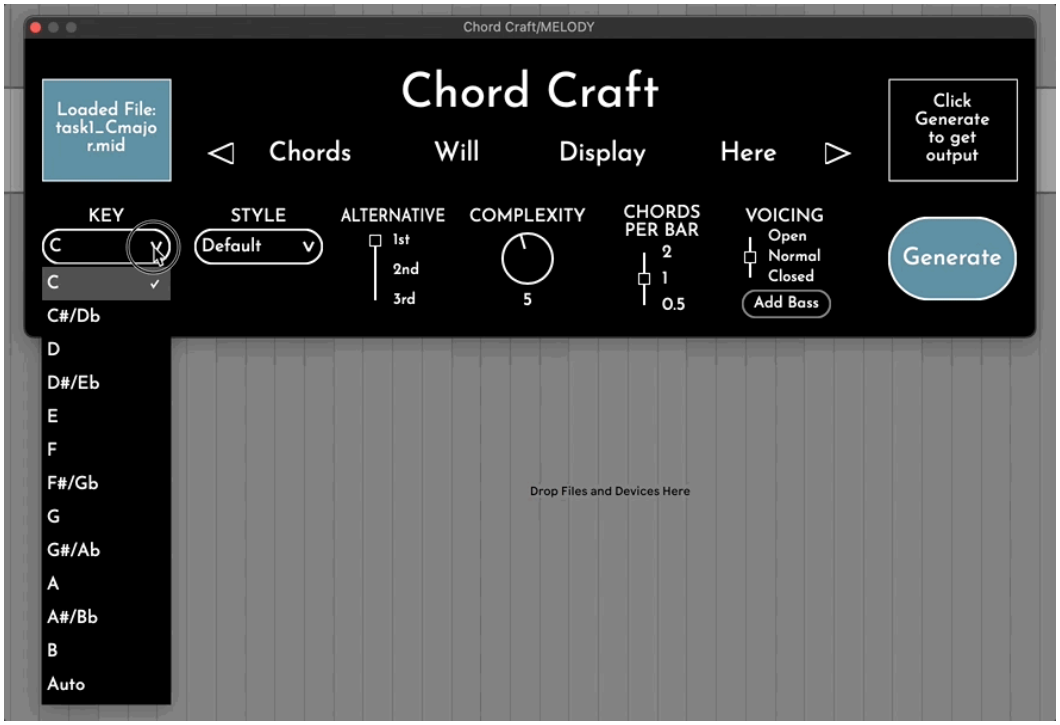


Figure 4.9: *Key and Tonality Parameters*

Then the user can quickly change multiple generation parameters at once by choosing a preset from the Style dropdown and adjust any other parameters to their liking (see Figure 4.10 below).

Figure 4.10: *Adjusting Parameters*

Once the desired parameter values are set, the Generate button must be clicked (see Figure 4.11 below).

Figure 4.11: *Generate Button*

Upon initiating chord generation via the "Generate" button, the MidiProcessor sends the input file along with all the current parameter values to our Python backend. This server processes the request, feeding the input file and parameters into our HMM chord generator to produce an output MIDI file with the generated chords, which is then sent back to the MidiProcessor. When the output file is ready, the output area is highlighted in green (see Figure 4.12 below).

Figure 4.12: *Output Ready*

Finally, now the user can drag the MIDI file containing the generated chords back into their DAW workspace for further use (see Figure 4.13 below).

Figure 4.13: *Dragging Output*

4.4.1 Example Generations

Below are two example chord progressions Chord Craft generated, the first using a melody in C Major (see Figures 4.14-16 and Table 4.3 below) and the other using a melody in G Minor (see Figures 4.17-19 and Table 4.4 below). Both were done using the Default preset settings, only the key and tonality parameters were changed to match the respective melodies. Each example has a video presenting the melody and the generated chord progression. The evaluation metrics for each generation are shown below as well.

4.4.2 Major Generation

C Major Example Generation: **Watch Video**

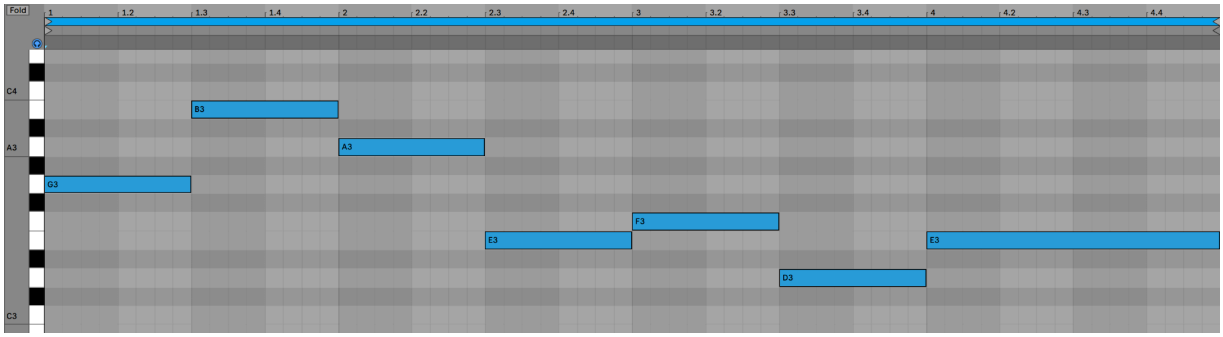


Figure 4.14: *Input melody file task1_Cmajor.mid*

Figure 4.15: *C Major generation settings*

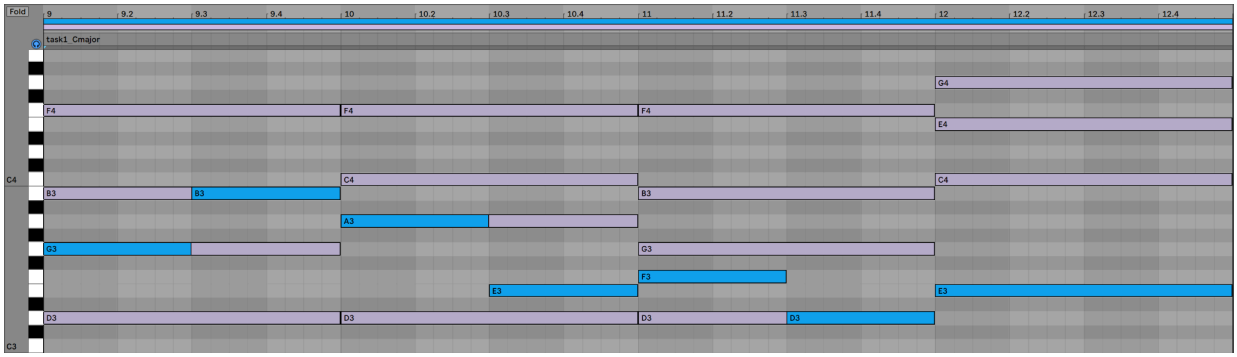


Figure 4.16: *Output C Major Chord Progression*

Table 4.3: C Major Generation Evaluation Metrics

Diatonicism	Note Fitting	Common Tones
100%	87.5%	39.6%

4.4.3 Minor Generation

G Minor Example Generation: **Watch Video**

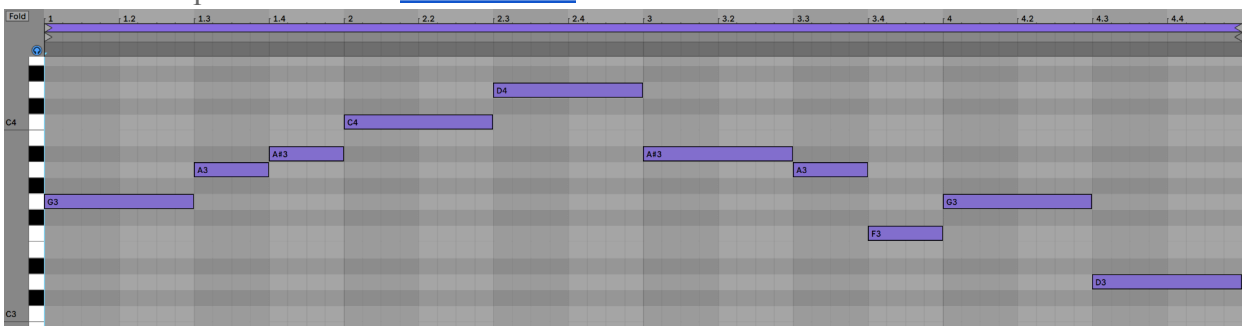


Figure 4.17: *Input melody task3_Gminor.mid*

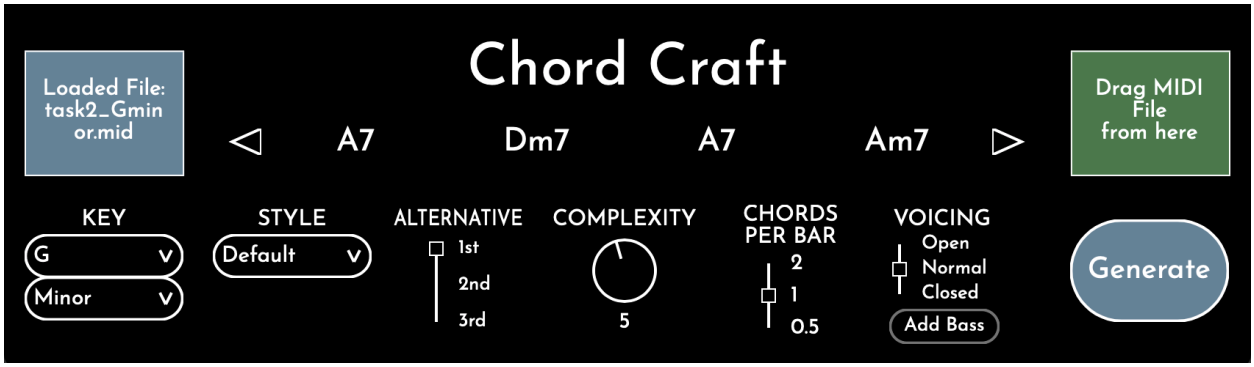


Figure 4.18: *G Minor Generation Settings*

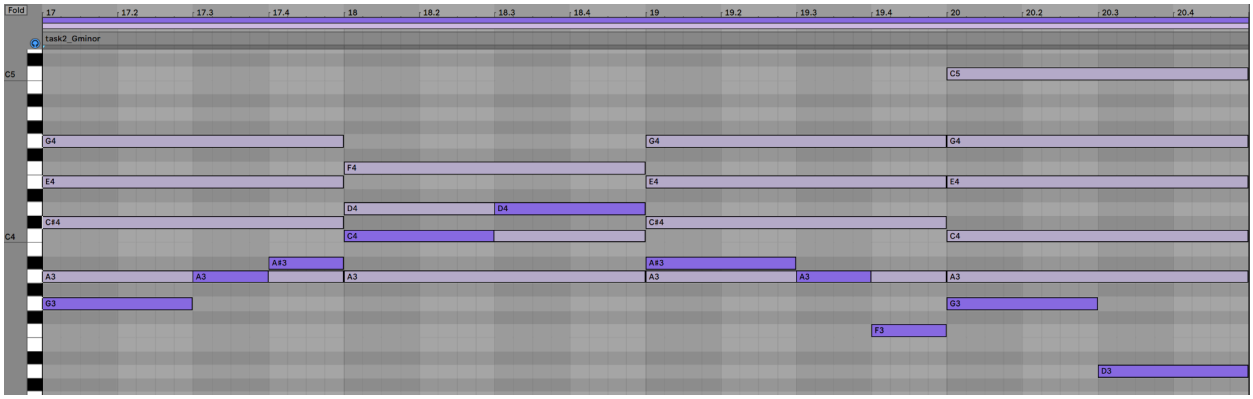


Figure 4.19: *Output G Minor Chord Progression*

Table 4.4: G Minor Generation Evaluation Metrics

Diatonicism	Note Fitting	Common Tones
67.7%	93.8%	58.3%

The metrics for the G minor generation provide useful insights into the success of ChordCraft. With a diatonicism calculation of 67.7%, it's evident that a good amount of the generated chords adhered to the G minor scale. However, roughly one third of the notes were not in the diatonic scale, suggesting that depending on the parameters, ChordCraft may decide to utilize accidentals. The high note fitting percentage of 93.8% indicates a strong alignment between the selected notes and the accompanying harmony. The common tones score of 48.3% suggests a balance between harmonic stability and variety.

4.5 Findings from User Study

Overall, users were satisfied with the output of the generated chord progressions. Users state that they would use the generated progressions as an inspirational starting point for the composition process.

For the simplest progressions generated, namely from Tasks 1 and 2 (Task 1 required the participant to create a chord progression and task 2 had the participant change three parameters), all users expressed a positive sentiment in their comments regarding the progressions.

Over the course of the study, 13 participants across varying musical backgrounds were surveyed. The demographics of participants varied greatly in terms of music composition experience and number of songs written, with 4/13 of participants stating they were "Beginner: Basic understanding with some informal exploration", 6/13 stating they were "Intermediate: Adequate knowledge with formal or informal training" and 3/13 stating that they were "Advanced: Proficient with significant formal training or professional experience".

Additionally, 4/13 of participants identified that this could be a valuable educational tool, helping beginners learn patterns in simple chord progressions against a melody.

Participants found all of the parameters useful.

For each of the statements regarding parameter usefulness (ie. I think the [parameter name] parameter is useful), the vast majority of participants responded with "strongly agree" and only one or two participants for each parameter responded with "slightly agree."

Users overall commented positively on the functionality, saying things such as “it’s a good way to generate chords a user might not expect or consider” and “I like it as an idea starter.” In fact, 9/13 participants explicitly stated they would use Chord Craft for getting harmonic inspiration.

Participants had an overall strong understanding of the influence of the parameters on the chord generation. However, some users demonstrated some confusion with the influence of the complexity and voicing parameters.

In general, participants had a strong understanding of the influence of parameters on the chord generation, regardless of their musical background. Participants strongly agreed with the statements “I understand how [parameter] parameter influences the chord generation” with regards to the key, chords per bar, bass, and tonality parameters.

Some participants found mild difficulty in understanding how the complexity, alternative and voicing parameters worked, with 3 participants rating the statement “I understand how the complexity parameter influences the chord generation” at slightly agree, 1 participant at neutral, while the remainder of the participants rated it at strongly agree.

Additionally, 3 participants rated the statement “I understand how the alternative parameter influences the chord generation” to slightly agree, while the remainder rated it strongly agree.

Users with varying musical backgrounds, namely in genre expertise, had varying opinions on similar generated chord progressions.

Many of the generations with a high complexity contained borrowed chords, a common occurrence in the jazz genre which features chords taken from a parallel key. This can sometimes create a very dissonant sound, which is another trademark of jazz. Notably, there were four users who encountered these chords in their progressions and made comments on them. Of the four , two mentioned a negative sentiment towards the borrowed chord. In one case, it was a A7 chord in the key of g minor, which in the context of the melody, created a chord which contained a #9, which is another common jazz idea. The second case was similar, having a G7 chord in the key of D major. The other two users had a positive sentiment towards the borrowed chords, with one

user experiencing the same #9 sound, albeit in a different melody in a different key. The two users who expressed positive sentiments had a jazz background, whereas the two who expressed a negative sentiment had a classical background. While this is only one example, it effectively demonstrates how users with varying genre backgrounds but similar levels of composing and music theory experience can have polarizing opinions on similar chord progressions.

5.0 Conclusion and Recommendations

As ChordCraft is a new software with much potential, we feel it's important to outline directions for the future. In this section, we provide recommendations for the continued development and validation of ChordCraft.

For future development, we recommend continuing to expand upon the available parameters to users. Since many of our participants exhibited different preferences for chord generations based on their musical genre expertise and the style of song they were composing, we can reasonably conclude that each user will approach using the software in a different way. Some example of parameters to add could be changes in the number of voices, which would allow someone who was writing music for a specific set of voices, such as in a choir, to easily hear the chords in a more accurate context or to allow the user to enable or disable the generation of specific chord types.

The style feature should be expanded upon to influence the generation to a greater extent than it currently does. One way to do this would be to train multiple HMMs on various datasets which are reduced to only songs of specific genres, and allow the user to specify which model they want to use based on which genre or style of song they wish to compose. Additionally, many participants expressed a desire for a greater level of customization over the harmonic rhythm of the generated chord progression. One potential feature which we recommend is the ability to visualize the melody within the VST and give the user the ability to place markers where they want the chords to be.

Suggestions for the UI included implementing a more dynamic color scheme, as ChordCraft's UI is mainly black and white. We also recommend the development of a loading bar during generation as the software often takes multiple seconds to generate a chord progression. The "Add Bass" button caused some confusion, prompting the consideration for renaming the button. A resizable window feature would also be beneficial as some users found the default window size to be too small. Lastly, a suggestion was made for truncating the file name on upload to avoid overflow, which indicated issues with how the interface displays and manages information.

For future studies, we propose several areas for improvement. These include conducting more extensive user testing to gather more feedback on usability and effectiveness, and finding a larger sample of individuals with a greater variance in musical ability to provide feedback on the software. While the initial sample provided us with valuable insights, expanding the sample pool would offer a more comprehensive understanding of ChordCraft's limitations and areas for improvement.

To ensure widespread accessibility of the software, our goal is to make ChordCraft commercially available to users as a packaged product. We would like to eliminate the need for individual setup of a Python server on each computer to make the VST easier to use and access. By packaging the software, we hope ChordCraft can reach a wider audience and reach its full potential among musicians looking to compose.

6.0 Glossary

6.1 Digital Music Production Terms

DAW: Digital Audio Workstations (DAWs), which were once defined as "[machines] equipped with sound cards and software for editing and processing digital audio (Leider, 2004)", the term in a modern context generally refers to "[any] software environments for music production (Marrington, 2017)." DAWs are used by music producers to record and edit audio, play virtual instruments, add audio effects, and to mix and master tracks. Some common DAWs which are commercially available are Ableton Live, Logic Pro, FL Studio, REAPER, Pro Tools, Audacity and Adobe Audition.

Plugin: Software that adds specific capabilities to a larger software application, commonly found in DAWs. Common formats include VST3 and AU.

MIDI: Music Instrument Digital Interface (MIDI), is a specification of a communications scheme between digital music devices (Loy, 1985). MIDI is the universal industry standard for electronic instruments and is a common language for musicians and DAWs to translate musical information in a digital space. Additionally, there are several music notation programs that allow a user to input traditional staff-based music notation and output MIDI such as Musescore.

VST: Virtual Studio Technology (VST) is an interface developed by Steinberg which aims to provide "seamless integration for virtual instruments and effects" (*Our Technologies*, n.d.). Any software program which takes advantage of this interface standard is commonly referred to as a VST and can be seamlessly integrated into a DAW workflow. Some examples of VST's include virtual instruments and MIDI effect producers.

AU: An Audio Unit (AU) is a plug-in architecture standard used in macOS for digital audio systems.

6.2 Software Development Terms

Xcode: A comprehensive IDE for macOS, supporting C++ and other languages, used primarily for developing applications for Apple ecosystems.

Visual Studio: A full-featured C++ IDE for Windows, supporting a wide range of software development projects.

REST API: A set of rules for how applications communicate over HTTP, facilitating interoperability between computer systems on the internet.

GUI: A Graphical User Interface (GUI) is a visual interface that allows users to interact with electronic devices through graphical icons and visual indicators.

CMake: An open-source, cross-platform family of tools designed to build, test, and package software primarily for C++.

JUCE Development Terms

JUCE: A widely used C++ framework for creating cross-platform software and plugins for digital audio workstations.

Projucer: A project management tool and code editor for JUCE, facilitating cross-platform development.

HMM: A Hidden Markov Model (HMM) is a statistical model used to represent systems that are assumed to be Markov processes with unobserved (hidden) states.

7.0 References

Absolu, B., Li, T., Ogihara, M. (2010). Analysis of Chord Progression Data. In: Raś, Z.W., Wieczorkowska, A.A. (eds) Advances in Music Information Retrieval. Studies in Computational Intelligence, vol 274. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-11674-2_8

Basu, S., Morris, D., & Simon, I. (2014). The songsmith story, or how a small-town hidden Markov model dade it to the big time. *The Journal of the Acoustical Society of America*, *135*(4), 2378–2378. https://doi.org/10.1121/1.4877851

Bharucha, J. J., & Krumhansl, C. L. (1983). The representation of harmonic structure in music: Hierarchies of stability as a function of context. *Cognition*, *13*(1), 63–102. https://doi.org/10.1016/0010-0277(83)90003-3

Chord Progression Generator—Inspiring Random Chords—SoundGrail. (n.d.). Retrieved October 11, 2023, from https://app.soundgrail.com/chord-progression-generator/

Chu, H., Kim, J., Kim, S., Lim, H., Lee, H., Jin, S., Lee, J., Kim, T., & Ko, S. (2022). An Empirical Study on How People Perceive AI-generated Music. *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. https://doi.org/10.1145/3511808.3557235

Daniel, J., & Martin, J. (2023). *Speech and Language Processing*. https://web.stanford.edu/~jurafsky/slp3/A.pdf

Software | Dmitri Tymoczko. (n.d.). Retrieved March 20, 2024, from https://dmitri.mycpanel.princeton.edu/software.html

Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, *61*(3), 268–278. https://doi.org/10.1109/PROC.1973.9030

Gibson, J. (n.d.). *The MIDI Standard: Introduction to MIDI and Computer Music: Center for Electronic and Computer Music: Jacobs School of Music*. Cecm.indiana.edu. https://cecm.indiana.edu/361/MIDI.html

Haerle, D. (1980). *The Jazz Language: A Theory Text for Jazz Composition and Improvisation: A Theory Text for Jazz Composition and Improvisation*. Alfred Music.

hmmlearn—Hmmlearn 0.3.2.post3+g97c2fc5 documentation. (n.d.). Retrieved March 20, 2024, from https://hmmlearn.readthedocs.io/en/latest/

HOME - chordprism. (n.d.). Www.chordprism.com. Retrieved March 14, 2024, from https://www.chordprism.com/

How's My Voice Leading? (n.d.). Greghmerrill.github.io. Retrieved March 14, 2024, from https://greghmerrill.github.io/voice-leading/

Juchniewicz, J., & Silverman, M. J. (2011). The influences of progression type and distortion on the perception of terminal power chords. *Psychology of Music*, *41*(1), 119–130. https://doi.org/10.1177/0305735611422506

Julia, J., Prana Dwija Iswara, & Tedi Supriyadi. (2021). Creating songs using online software. *Journal of Physics: Conference Series*, *1987*(1), 012018–012018. https://doi.org/10.1088/1742-6596/1987/1/012018

Kulesz, O. (2018). *Culture, platforms and machines:the impact of artificial intelligence on the diversity of cultural expressions* (p. 17). Intergovernmental Committee for the Protection and PROMOTION of the Diversity of Cultural Expressions

https://www.academia.edu/37933741/Culture_platforms_and_machines?auto=download&campaign=weekly_digest

Lead Sheet | Berklee College of Music. (n.d.). College.berklee.edu. Retrieved March 13, 2024, from https://college.berklee.edu/bt/193/leadsheet.html

Lee, J., & Lee, J.-S. (2018). Music Popularity: Metrics, Characteristics, and Audio-Based Prediction. *IEEE Transactions on Multimedia*, *20*(11), 3173–3182. https://doi.org/10.1109/tmm.2018.2820903

Li, T., Choi, M., Fu, K., & Lin, L. (2019, December 1). *Music Sequence Prediction with Mixture Hidden Markov Models*. IEEE Xplore. https://doi.org/10.1109/BigData47090.2019.9005695

Martin, S. (1999). *Designing Effective User Interfaces*. Retrieved October 10, 2023, from https://web.cs.wpi.edu/~matt/courses/cs563/talks/smartin/int_design.html

McKinney, J. (n.d.). *Research Guides: Music Notation: Preferred Preservation Formats for Digital Scores: MusicXML*. Guides.loc.gov. Retrieved February 28, 2024, from https://guides.loc.gov/music-notation-preferred-preservation-formats-for-digital-scores/musicxml

Mortimer, J. H., Nosko, C., & Sorensen, A. (2014, March). Supply responses to digital distribution: Recorded music and live performances. *ScienceDirect*, *24*(1), 3-14. https://doi.org/10.1016/j.infoecopol.2012.01.007

music21: A Toolkit for Computer-Aided Musicology. (n.d.). Retrieved March 20, 2024, from https://web.mit.edu/music21/

MusicXML 4.0. (n.d.). Retrieved March 20, 2024, from https://www.w3.org/2021/06/musicxml40/

Nettles, B. *(1997). The Chord Scale Theory & Jazz Harmony. Advance Music,.*

Norberg, K., & Norell, O. (2023). The Influence of Artificial Intelligence on Songwriting: Navigating Attribution Challenges and Copyright Protection. https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-334274

Nielsen, J. (2012, January 15). *Thinking Aloud: The #1 Usability Tool*. Nielsen Norman Group. https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/

Onerpmny. (2022, April 12). *Understanding VST's, Plugins, and DAW's*. ONErpm Blog. https://blog.onerpm.com/tips-tricks/vsts-plugins-and-daws/

Simonetta, F. (2024). *00sapo/OpenEWLD* [Python]. https://github.com/00sapo/OpenEWLD (Original work published 2018)

shiehn/chord-melody-dataset: This is a dataset to dedicated to the relationship between chords & melodies. (n.d.). Retrieved March 20, 2024, from https://github.com/shiehn/chord-melody-dataset

Songsmith. (n.d.). Microsoft Research. https://www.microsoft.com/en-us/research/project/songsmith-2/

Pachet, F., & Roy, P. (2001). Musical harmonization with constraints: A survey. *Constraints*, *6*, 7-19.

Peters, G. D. (1992). Music Software and Emerging Technology: G. David Peters outlines the history of music software and hardware and explores the new developments and benefits of the emerging software for use in the classroom. Music Educators Journal, 79(3), 22-63. https://doi.org/10.2307/3398478

Tamposis, I. A., Tsirigos, K. D., Theodoropoulou, M. C., Kontou, P. I., & Bagos, P. G. (2019). Semi-supervised learning of Hidden Markov Models for biological sequence analysis. *Bioinformatics (Oxford, England)*, *35*(13), 2208–2215. https://doi.org/10.1093/bioinformatics/bty910

Use Drummer in GarageBand for iPad. (n.d.). Apple Support. Retrieved March 13, 2024, from https://support.apple.com/guide/garageband-ipad/use-drummer-chs9692b2133/ipados

Valhalla Supermassive. (n.d.). Valhalla DSP. https://valhalladsp.com/shop/reverb/valhalla-supermassive/

hmmlearn—Hmmlearn 0.3.2.post3+g97c2fc5 documentation. (n.d.). Retrieved March 20, 2024, from https://hmmlearn.readthedocs.io/en/latest/

Why Lead Sheets? | Berklee. (n.d.). Www.berklee.edu. https://www.berklee.edu/berklee-today/summer-2018/lead-sheet

Woodruff, H. E. (2010). *Woodruff's Comprehensive Music Course*. Kessinger Publishing.

Yin, Z., Reuben, F., Stepney, S., & Collins, T. (2021). "A Good Algorithm Does Not Steal – It Imitates": The Originality Report as a Means of Measuring When a Music Generation

Algorithm Copies Too Much. In J. Romero, T. Martins, & N. Rodríguez-Fernández (Eds.), *Artificial Intelligence in Music, Sound, Art and Design* (pp. 360–375). Springer International Publishing. https://doi.org/10.1007/978-3-030-72914-1_24

Zplane. (2012). *Vielklang Instant Harmony 2 - zplane plug-ins*. Vielklang 2 Manual. https://products.zplane.de/wp-content/downloads/vielklang/vielklang2_manual.pdf

Zplane. (2012). *vielklang 2 tutorial 1: introduction*. *YouTube*. Retrieved October 10, 2023, from https://youtu.be/D0OC-K42z-E?si=rTbYrhVHa2Kx4SGI&t=112.

Appendix A: JUCE Development Resources

This appendix lists the JUCE resources utilized during the project that guided various aspects of the development process.

Videos

- Projucer Setup & Opening Demo Runner: Watch Video
- Overview of a Typical JUCE Audio Plugin: Watch Video
- Basic Gain Slider Audio Plugin Tutorial (Shows Xcode build scheme to open DAW automatically): Watch Video
- MIDI Plugin Tutorial: Watch Video
- MIDI File Loader Tutorial: Watch Video
- Drag and Drop Files into Plugin Window Tutorial: Watch Video

JUCE Website Tutorials

- JUCE Tutorials: Link
- Using Projucer Tutorial: Link
- Customizing Look and Feel: Link

Helpful JUCE Documentation

- JUCE Documentation: Link
- Drag Input Files into Plugin: Link
- External Dragging of Files from Plugin out to Computer: Link
- Xcode 15 Linker Issue: Link
- Sending POST Request with JUCE: Link

Example Open Source JUCE Plugins

- Ripchord: Link
- List of Plugins + Tools: Link

CMake Exploration

- Pamplejuce Github: Link
- Pamplejuce Overview Article: Link
- JUCE CMake API: Link
- Wolfsound CMake: Watch Video
- Wolfsound Github Template: Link

Appendix B: User Personas

Persona 1: Jake - Hobbyist Producer

Demographic:
Jake is a 23 year old college graduate with a degree in computer science, and has a passion for music.

Goals and Motivations:
Wants to produce unique tracks quickly. Seeks tools that can assist and inspire him in the music-making process. Motivated by hearing the end result and sharing it with friends.

Musical Background:
No formal music theory training. Most of what they know about music composition comes from introductory Ableton tutorials on YouTube and playing saxophone in the middle school jazz band.

Technological Background:
Very comfortable with DAWs and different VST plugins. Often experiments with new music tech tools.

Pain Points:
Struggles with music theory, specifically in building chord progressions. Often feels creatively stuck and needs inspiration.

Scenario:
Jake has just written a catchy melody that he thinks could be the hook of his new track. He's unsure what chords would fit underneath it. He's tried a few progressions, but they aren’t feeling right. He remembers he has a new VST plugin that generates chord progressions based on melodies. He decides to give it a try.

Jake Stories:

Story 1

As Jake, I want the VST to quickly analyze my melody so that I can receive chord progression suggestions immediately.

Acceptance Criteria:

- The VST should provide a clear interface to input or drag-and-drop a MIDI melody.
- The analysis should take no longer than a few seconds.
- Multiple chord progression suggestions should be presented.

Story 2

As Jake, I want the ability to tweak and modify the suggested progressions so that I can personalize them to my taste.

Acceptance Criteria:

- There should be options to change individual chords within the progression.
- The VST should offer alternative chords that also fit the melody.
- There should be an option to change the complexity or mood of the entire chord progression.

Persona 2: Amy - The Music Teacher

Background and Demographics:

Age: 35

Lives in a small town

Music teacher at the local high school

Goals and Motivations:

Wants to have multiple options of chord progressions to show her students. She wants to emphasize that there are often multiple correct chord progressions for one given melody. The students will not use the software, but she uses it for instructional purposes only.

Pain Points:

Is struggling to figure out how to teach chord progressions in an organized lesson format to her students and explain why certain chord progressions work better than others.

Technological Competence:

Has minimal DAW experience but found this plugin from googling automatic chord generators.

Scenario for Amy:

Amy is preparing lesson plans for her upcoming classes. She wants to demonstrate how melodies can be harmonized differently. Using the VST plugin, she hopes to show her students various ways a melody can be backed by chords.

Story 1:

Amy wants to input a simple student-composed melody into the VST so that I can show multiple harmonization options in class.

Acceptance criteria:

- The VST should allow for directly inputting a melody
- The VST should output multiple chord progression options

Story 2:

Amy wants to input some popular pop melodies and look at the different options of chord progressions to show the class.

Acceptance criteria:

- The VST should have an import feature for the melody
- The VST should display the chord progressions in an easy-to-read manner

Persona 3: John, the professional film score composer
Age: 65

John lives in Los Angeles and has been writing scores for short films for the past 30 years.

Goals and motivations:
As a long time user of traditional music notation software, John is looking to expand the tools he uses in composing for short films by using a DAW.

Pain Points:
John is often contracted to compose music for short film scenes with a very quick turnaround time. Oftentimes, he finds himself stressed about finding novel chord progressions for new melodies in that short period of time. For his past three projects, he's been harmonizing all of his tracks with very similar progressions, which has led them to sound similar unintentionally.

Technological competence:

John is a beginner user of DAWs and plug-ins. Most of what he's learned comes from YouTube and casually talking to a digital music producing colleague.

Musical background:
John has been composing for many different ensembles for the past 30 years, including orchestra, big band, string quartet and choir. He has a PhD in classical music composition from Julliard and is very familiar with traditional music notation and music theory. However, he is not very up to date on modern music styles such as EDM and pop.

Scenario:
John has only one week to compose and record a score for a 20 minute short film. The short film has two main characters, each with a distinct character motif that comes in the form of two four-bar melodies. Because both of the characters undergo a significant change in the film, John wants to illustrate that by having their melodies played several times but with different harmonies. To find inspiration for these different chord progressions, John turns to this VST.

Story 1:
As John, I want to have several chord progressions generated for a single input melody so that I can draw inspiration from those options in the composition process.

Story 2:
As John, I want to be able to have chord progressions generated in modern music styles such as EDM and pop so that I can create a more modern sound.

Persona 4: Sarah - Studio Session Producer

Demographic:

Sarah is a 24 year old with a degree in audio engineering, and recently got hired to work in a studio to help lyricists who pay for recording sessions.

Goals and Motivations:
Wants to be able to come up with chord progressions on the fly during a session. Motivated by satisfying the client artist with the end product.

Musical Background:
Has formal music theory training through school, but theory/composition was not her main focus.

Technological Background:
Years of DAW experience, extremely knowledgable about sound design.

Pain Points:
Becomes anxious when working under high-pressure scenarios. Prefers to take her time, so she struggles during sessions that have short time constraints, especially when it comes to the composition side compared to mixing/mastering.

Scenario:

A client comes in for a two-hour session, and all they have prepared is their lyrics and a melody. The client expects to leave the session with a full song. Sarah is worried that she won't be able to arrange the harmony in due time, and would rather focus more on the production. Sarah hopes that her new VST plugin will speed up the writing process, leaving more time for her to build a solid mix.

Story 1

Sarah wants to have substitution options for generated chords in case the client wants to make a quick change to the chord progression.

Acceptance Criteria:

- The VST, after having generated a progression, should allow the user to select certain chords, and suggest alternative chords that could also function in that slot.
- For any given chord, there should be an ample amount of substitution suggestions provided, but also not so many that it invokes decision paralysis
- It should be as easy as clicking a chord and picking a new one to take its place from a dropdown menu- a change that can be made in seconds.

Story 2

Sarah wants different genres of generation to select from based on her client's desires, so that she doesn't need to spend as much time coming up with basic styling herself.

Acceptance Criteria:

- The VST should have a variety of genres to pick from, i.e. jazz, rock, trap, that have their own distinct and accurate harmonic tendencies.

Persona 5: Raj - DJ and Electronic Music Producer

Demographic:

Raj is a 30 year old DJ and electronic music producer who frequently performs at clubs and is known for producing catchy remixes.

Goals and Motivations:

Aims to create remixes that can captivate the club audiences. He is constantly in search of VST tools and software to help create remixes of new songs.

Musical Background:

Raj has learned music theory by producing digital music for 10 years and is well versed in creating many genres of electronic dance music.

Technological Background:
Very comfortable with DAWs, VST plugins, and DJ software. Often experiments with new music tech tools.

Pain Points:
Raj often finds himself tasked with writing remixes for several new popular songs each week and can sometimes find it difficult to come up with new chord progressions for these popular melodies quickly, as he prides himself on creating unique, catchy remixes of popular songs.

Scenario:
Raj is currently working on remixing a classic pop hit. While the melody is iconic, he wants to reharmonize it and bring new flavor to the remix. Raj is looking for a VST that can suggest unconventional chord progressions to complement the melody.

Raj Stories:

Story 1
As Raj, I want the VST to provide unique and non-traditional chord progression suggestions for the classic melody so that I can craft a remix that stands out.

Acceptance Criteria:

- The VST must allow easy input or drag-and-drop of the MIDI melody.
- Suggestions should not just stick to conventional progressions but offer some avant-garde choices.
- The VST should present a range of progressions from simple to complex.

Story 2

As Raj, I want the ability to customize the chord progressions so that I can align them with the mood and vibe I'm aiming for in my remix.

Acceptance Criteria:

- The VST should offer in-depth customization options for each chord in the progression.
- There should be a feature to change the mood, tension, or color of the suggested progressions.
- The VST should allow Raj to merge or blend multiple progression suggestions.

Appendix C: User Instructions

Overview

There will be four parts which should take no longer than an hour total.

1. Intro Survey (5 min)
2. Demo Video (5 min)
3. Activity (30 min)
4. Final Survey (5 min)

Intro Survey

First complete our pre-testing Qualtrics Survey

Survey Link: https://wpi.qualtrics.com/jfe/form/SV_3PfQrKfVZljDMLs

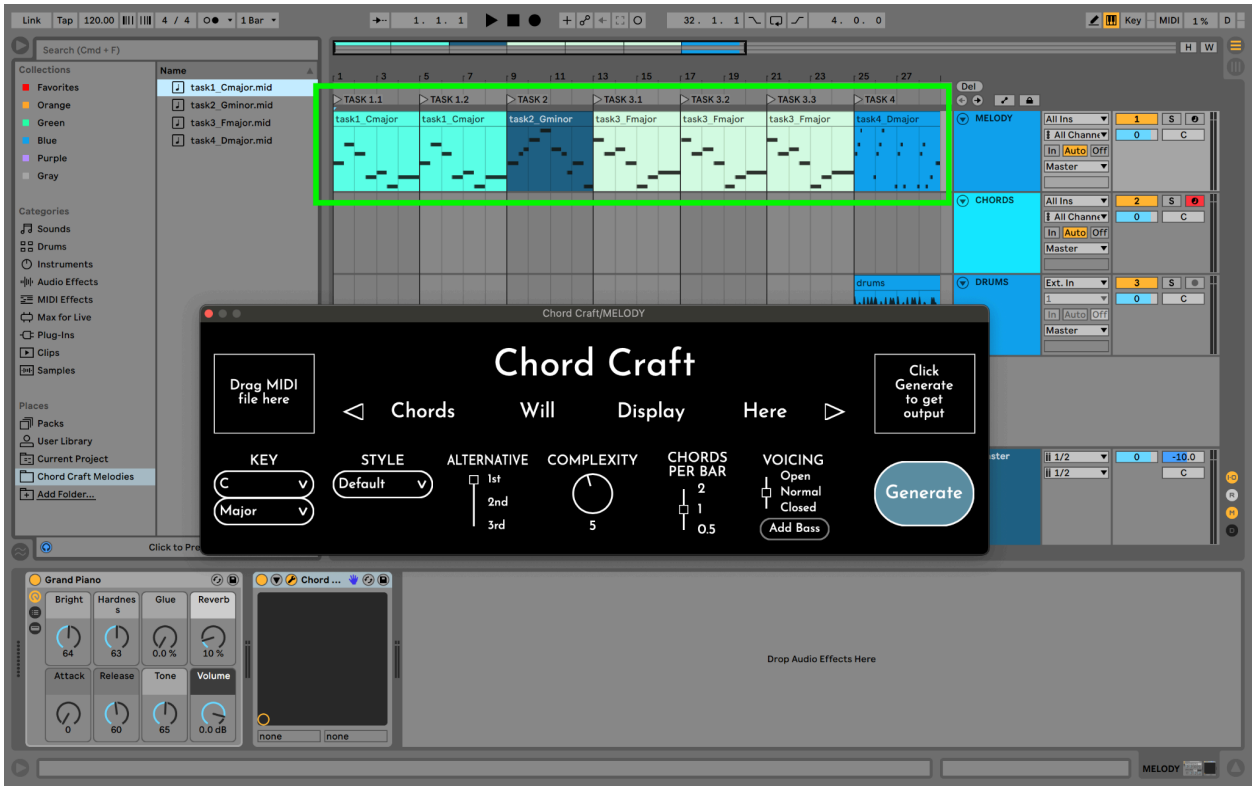
Video

Watch the demonstration video showing how to use our software.

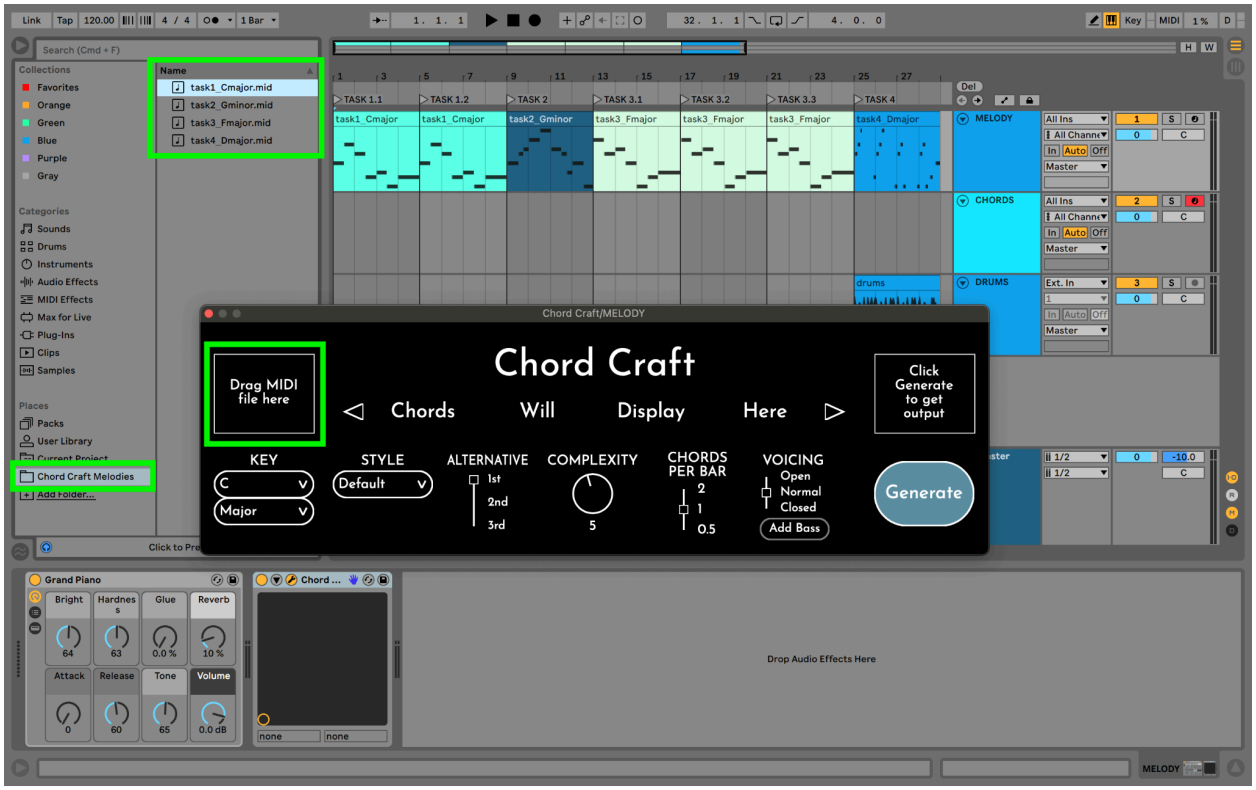
Video Link: https://youtu.be/KjUeHb2rjW8?si=lRp2XVqhq739PCrV

Activity

You will be testing our software which is a Virtual Studio Technology (VST) plugin in Ableton Live, a popular Digital Audio Workstation. You will complete four tasks usings MIDI files containing melodies. The Ableton project has been set up with the correct melodies arranged at each task location.



For each task, you will need to drag the correct melody file into Chord Craft's input area. The MIDI files for each task are located in the Chord Craft Melodies folder in Ableton's File Explorer.



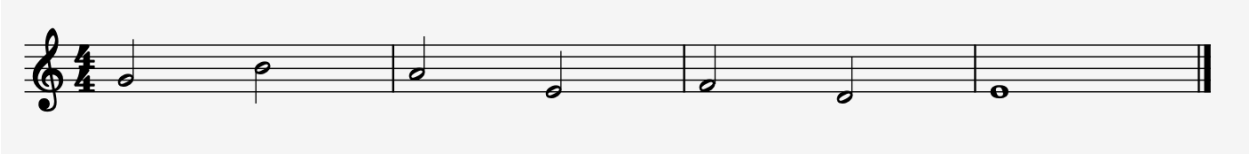
Now using the software, complete each of the following tasks at their correct location in the Ableton project. As you complete the tasks, please share your thoughts with the interviewer with regards to ease of use, satisfaction with the output chord progression, etc.

The Melody track is loaded with a grand piano sound and our VST Chord Craft.

The Chords track is loaded with an electric piano sound.

Task 1.1

Use task1_Cmajor.mid to create a simple chord progression with one chord per bar. Below is the task1 MIDI melody in notation form.

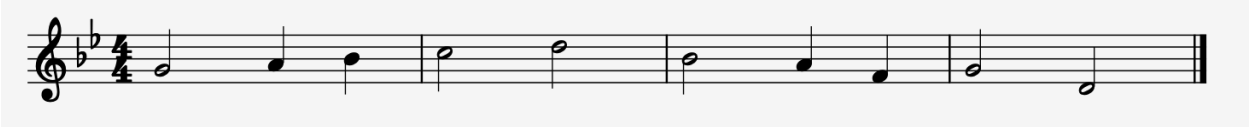


Task 1.2

Using the same task1_Cmajor.mid melody, adjust at least three of the parameters to generate a new chord progression.

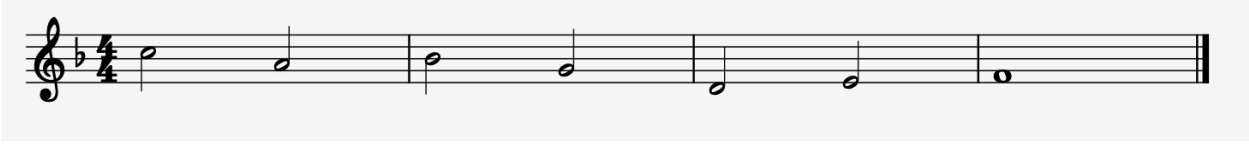
Task 2

Now use task2_Gminor.mid to create a chord progression in G minor that has two chords per bar. Below is the task2 MIDI melody in notation form.



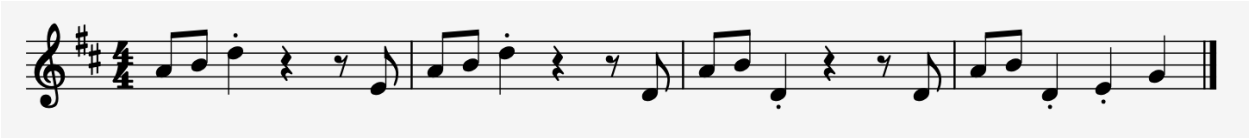
Task 3.1- 3.3

For each melody and set of parameters, there are three possible chord progressions generated. You can select a progression using the alternative parameter. For this task, use task3_Fmajor.mid. Choose any series of parameters and drag each of the three progressions at each task location in Ableton. Compare and describe your thoughts on each of the progressions. Below is the task3 MIDI melody in notation form.



Task 4

Listen to the task4_Dmajor.mid melody playing along with the drum beat. Use Chord Craft to generate a chord progression that fits the song.



Final Survey

Please fill the remaining section of our survey after completing the activity.

Appendix D: User Survey Results

Page 1

Chord Craft User Study

Mar 21, 2024 1:35 AM

How many years of formal music theory education do you have?

2

5

7.5

1

2

2

Do you play any instruments / sing? If so, which instruments and how many formal years of experience do you have (ie. performing with an ensemble, taking lessons)?

violin - 10 year of lessons, bass 30 years self taught, have played in ensembles for 40 years

Trumpet (9 years), piano (5 years), drum (6 years)

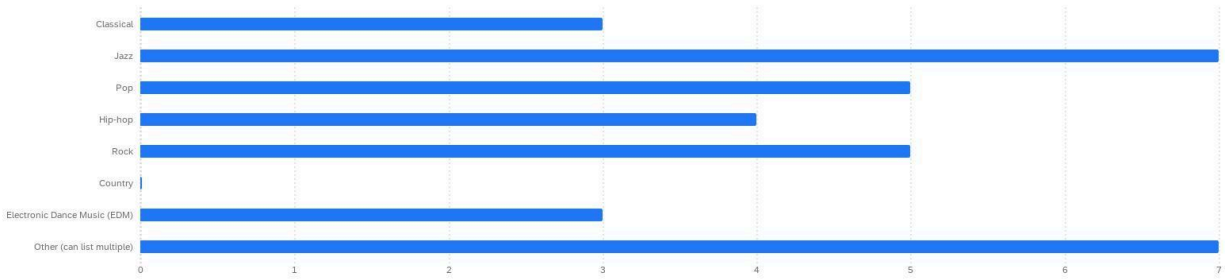
I play guitar, sing in A Cappella/Chior. 12.5 years since I started playing Guitar.

Tenor sax: 7 years; Singing: 3 years

Cello 14 years, Singing 15 years. I have experience with youth ensembles, lessons, and wpi ensembles

I play trombone and French horn

Which of the following genres of music do you regularly listen to? 13



Which of the following genres of music do you regularly listen to? 13

Q3 - Which of the following genres of music do you regularly listen to? - Selected Choice	Percentage	Count
Classical	23%	3
Jazz	54%	7
Pop	38%	5
Hip-hop	31%	4
Rock	38%	5
Country	0%	0
Electronic Dance Music (EDM)	23%	3

Which of the following genres of music do you regularly listen to?: Other (can list multiple)

funk, jazz fusion, reggae, world

Funk, Fusion

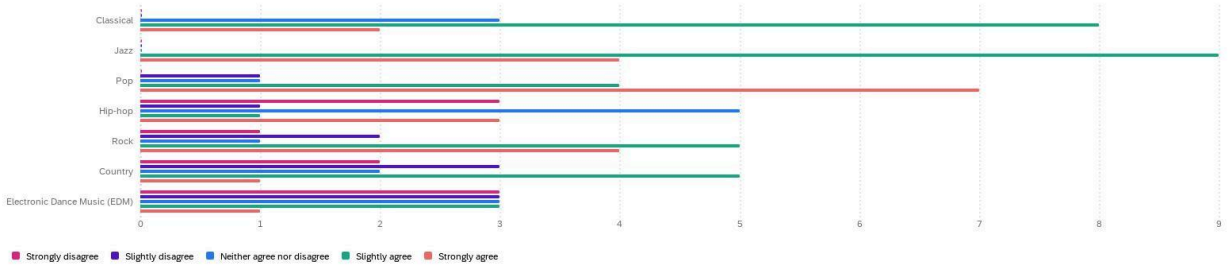
Funk

Jazz Fusion

granola

jazz, hip hop, soul, rnb, funk, rock, video game/tv/film soundtracks, neo soul, edm, dance, house, reggaeton, brazilian samba, brazilian pagode

For each of the following genres of music, choose the statement that best reflects your agreement with the following statement: I understand the music theory (such as structure, common chord progressions, motifs, typical instrumentation, etc.) associated with this genre. 13



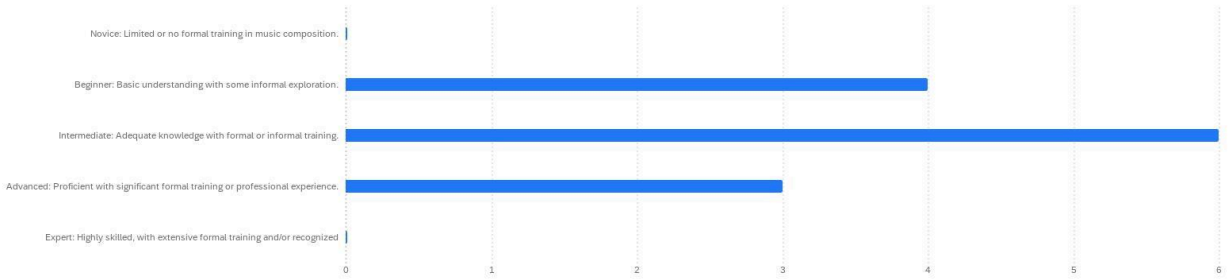
For each of the following genres of music, choose the statement that best reflects your agreement with the following statement: I understand the music theory (such as structure, common chord progressions, motifs, typical instrumentation, etc.) associated with this genre. 13

For each of the following genres of music, choose the statement that best r... ↑	Strongly disagree	Slightly disagree	Neither agree nor disagree	Slightly agree	Strongly agree
Classical	0	0	3	8	2
Jazz	0	0	0	9	4
Pop	0	1	1	4	7
Hip-hop	3	1	5	1	3
Rock	1	2	1	5	4
Country	2	3	2	5	1
Electronic Dance Music (EDM)	3	3	3	3	1

For each of the following genres of music, choose the statement that best reflects your agreement with the following statement: I understand the music theory (such as structure, common chord progressions, motifs, typical instrumentation, etc.) associated with this genre. 13

For each of the following genres of music, choose the statement that best r...	Average	Minimum	Maximum	Count
Classical	3.92	3.00	5.00	13
Jazz	4.31	4.00	5.00	13
Pop	4.31	2.00	5.00	13
Hip-hop	3.00	1.00	5.00	13
Rock	3.69	1.00	5.00	13
Country	3.00	1.00	5.00	13
Electronic Dance Music (EDM)	2.69	1.00	5.00	13

Considering your experience in music composition, please select the option that best describes your background: 13



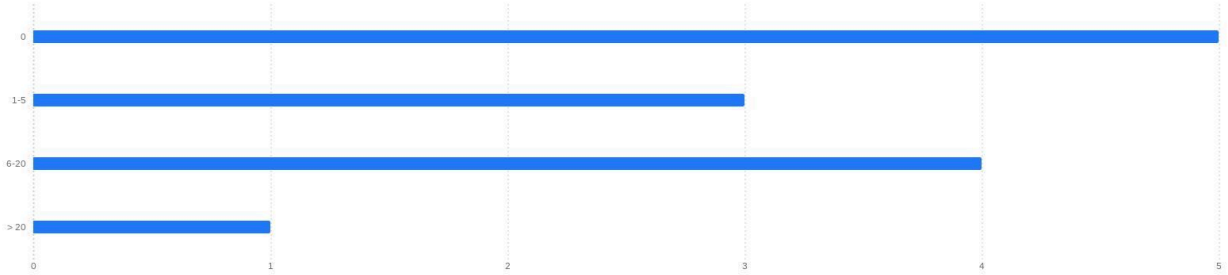
Considering your experience in music composition, please select the option that best describes your background: 13

Q5 - Considering your experience in music composition, please select the option that best describes your background:	Percentage	Count
Novice: Limited or no formal training in music composition.	0%	0
Beginner: Basic understanding with some informal exploration.	31%	4
Intermediate: Adequate knowledge with formal or informal training.	46%	6
Advanced: Proficient with significant formal training or professional experience.	23%	3
Expert: Highly skilled, with extensive formal training and/or recognized	0%	0

Considering your experience in music composition, please select the option that best describes your background: 13

Considering your experience in music composition, please select the option...	Average	Minimum	Maximum	Count
Novice: Limited or no formal training in music composition.	-	-	-	0
Beginner: Basic understanding with some informal exploration.	2.00	2.00	2.00	4
Intermediate: Adequate knowledge with formal or informal training.	3.00	3.00	3.00	6
Advanced: Proficient with significant formal training or professional experience.	4.00	4.00	4.00	3
Expert: Highly skilled, with extensive formal training and/or recognized	-	-	-	0

How many songs have you written? 13



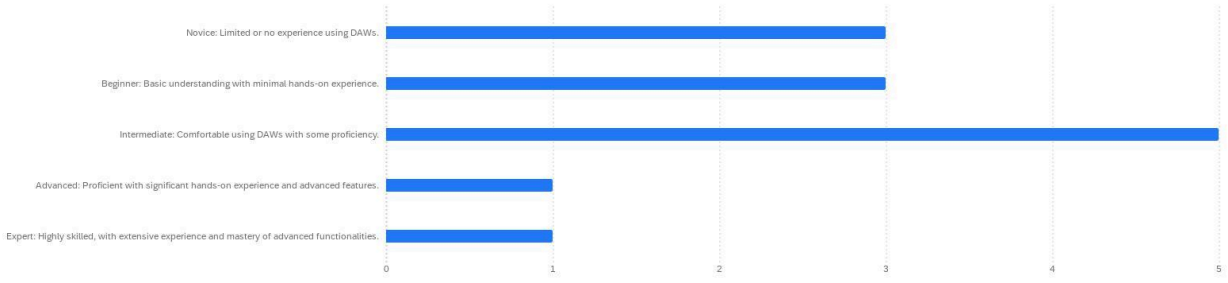
How many songs have you written? 13

Q6 - How many songs have you written?	Percentage	Count
0	38%	5
1-5	23%	3
6-20	31%	4
> 20	8%	1

How many songs have you written? 13

How many songs have you written?	Average	Minimum	Maximum	Count
0	1.00	1.00	1.00	5
1-5	2.00	2.00	2.00	3
6-20	3.00	3.00	3.00	4
> 20	4.00	4.00	4.00	1

Considering your experience with Digital Audio Workstations (DAWs), please select the option that best describes your background: 13



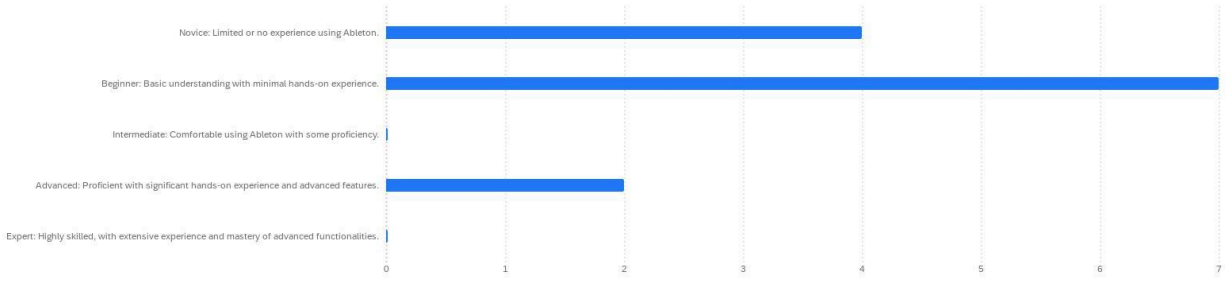
Considering your experience with Digital Audio Workstations (DAWs), please select the option that best describes your background: 13

Q7 - Considering your experience with Digital Audio Workstations (DAWs), please select the option that best describes your background:	Percentage	Count
Novice: Limited or no experience using DAWs.	23%	3
Beginner: Basic understanding with minimal hands-on experience.	23%	3
Intermediate: Comfortable using DAWs with some proficiency.	38%	5
Advanced: Proficient with significant hands-on experience and advanced features.	8%	1
Expert: Highly skilled, with extensive experience and mastery of advanced functionalities.	8%	1

Considering your experience with Digital Audio Workstations (DAWs), please select the option that best describes your background: 13

Considering your experience with Digital Audio Workstations (DAWs), please...	Average	Minimum	Maximum	Count
Novice: Limited or no experience using DAWs.	1.00	1.00	1.00	3
Beginner: Basic understanding with minimal hands-on experience.	2.00	2.00	2.00	3
Intermediate: Comfortable using DAWs with some proficiency.	3.00	3.00	3.00	5
Advanced: Proficient with significant hands-on experience and advanced features.	4.00	4.00	4.00	1
Expert: Highly skilled, with extensive experience and mastery of advanced functionalities.	5.00	5.00	5.00	1

Considering your experience with Ableton Live, please select the option that best describes your background: 13



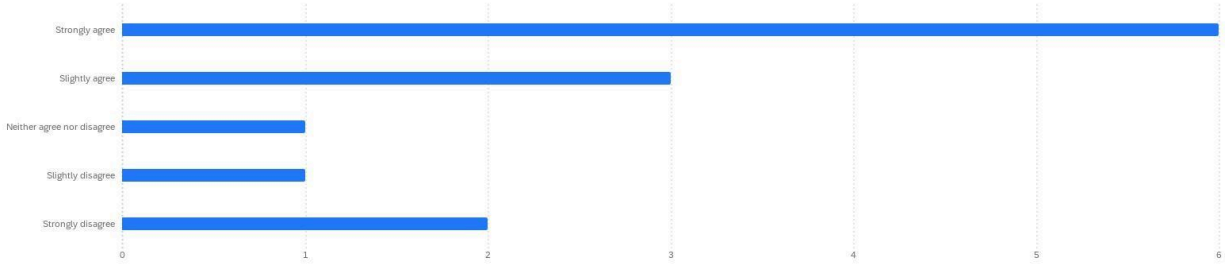
Considering your experience with Ableton Live, please select the option that best describes your background: 13

Q8 - Considering your experience with Ableton Live, please select the option that best describes your background:	Percentage	Count
Novice: Limited or no experience using Ableton.	31%	4
Beginner: Basic understanding with minimal hands-on experience.	54%	7
Intermediate: Comfortable using Ableton with some proficiency.	0%	0
Advanced: Proficient with significant hands-on experience and advanced features.	15%	2
Expert: Highly skilled, with extensive experience and mastery of advanced functionalities.	0%	0

Considering your experience with Ableton Live, please select the option that best describes your background: 13

Considering your experience with Ableton Live, please select the option tha...	Average	Minimum	Maximum	Count
Novice: Limited or no experience using Ableton.	1.00	1.00	1.00	4
Beginner: Basic understanding with minimal hands-on experience.	2.00	2.00	2.00	7
Intermediate: Comfortable using Ableton with some proficiency.	-	-	-	0
Advanced: Proficient with significant hands-on experience and advanced features.	4.00	4.00	4.00	2
Expert: Highly skilled, with extensive experience and mastery of advanced functionalities.	-	-	-	0

Considering your experience in music composition and performance, choose the option that best reflects your agreement with the following statement: Given a simple diatonic melody, I am able to write a chord progression which "fits" that melody. 13



Considering your experience in music composition and performance, choose the option that best reflects your agreement with the following statement: Given a simple diatonic melody, I am able to write a chord progression which "fits" that melody. 13

Q9 - Considering your experience in music composition and performance, choose the option that best reflects your agreement with the following statement: Given a simple diatonic melody, I am able to write a chord progression which "fits" that melody.	Percentage	Count
Strongly agree	46%	6
Slightly agree	23%	3
Neither agree nor disagree	8%	1
Slightly disagree	8%	1
Strongly disagree	15%	2

Considering your experience in music composition and performance, choose the option that best reflects your agreement with the following statement: Given a simple diatonic melody, I am able to write a chord progression which "fits" that melody. 13

Considering your experience in music composition and performance, choose th...	Average	Minimum	Maximum	Count
Strongly agree	1.00	1.00	1.00	6
Slightly agree	2.00	2.00	2.00	3
Neither agree nor disagree	3.00	3.00	3.00	1
Slightly disagree	4.00	4.00	4.00	1
Strongly disagree	5.00	5.00	5.00	2

What factors would you consider when choosing chords for a progression?

which are the strong notes in the melody and what triads are they part of

Creating a structure that leads itself, following typical principles of dominant/subdominant/etc chords, adding interest with extensions or substitutions, etc

They need to be in the same key of course. It has to sound good.

Major, minor, augmented, diminished; desired key

Key of Music, the tone that you want to promote within the progression

Avoid tones

Our project is a piece of software in the form of a Virtual Studio Technology (VST) plug-in for a DAW. It has the ability to take a melody in MIDI format combined with a series of parameters as an input and outputs a chord progression using machine learning. What initial thoughts do you have on the potential use cases and features that this software could have?

sounds cool for people like me with lots of interest in making music but limited knowledge of harmony

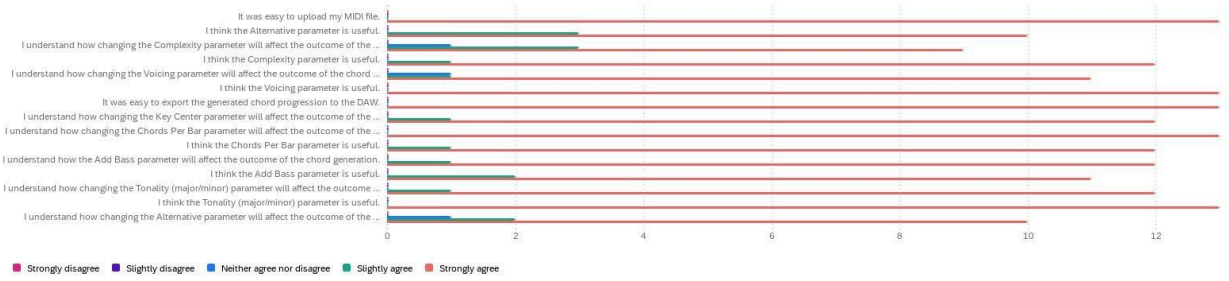
It could be used to help those who are not well versed in music theory to create their music. Depending on its accuracy, it could also be used as an academic tool to help students learn or check their work when in lower-level theory classes.

I'm thinking about how innovative this sounds as it can help ease the process of music production.

Could be very useful in writing music in general. Could also be used as an educational tool, being able to be given a chord and then say what type it is (this could also be used in composing as well).

This could assist in promoting ideas surrounding different chord progressions

Based on your overall experience using the plugin, rate how much you agree with the following statement. 13



Based on your overall experience using the plugin, rate how much you agree with the following statement. 13

Based on your overall experience using the plugin, rate how much you agree... ↑	Strongly disagree	Slightly disagree	Neither agree nor disagree	Slightly agree	Strongly agree
It was easy to upload my MIDI file.	0	0	0	0	13
I think the Alternative parameter is useful.	0	0	0	3	10
I understand how changing the Complexity parameter will affect the outcome of the chord generation.	0	0	1	3	9
I think the Complexity parameter is useful.	0	0	0	1	12
I understand how changing the Voicing parameter will affect the outcome of the chord generation.	0	0	1	1	11
I think the Voicing parameter is useful.	0	0	0	0	13

Based on your overall experience using the plugin, rate how much you agree with the following statement. 13

Based on your overall experience using the plugin, rate how much you agree...	Average	Minimum	Maximum	Count
It was easy to upload my MIDI file.	5.00	5.00	5.00	13
I think the Alternative parameter is useful.	4.77	4.00	5.00	13
I understand how changing the Complexity parameter will affect the outcome of the chord generation.	4.62	3.00	5.00	13
I think the Complexity parameter is useful.	4.92	4.00	5.00	13
I understand how changing the Voicing parameter will affect the outcome of the chord generation.	4.77	3.00	5.00	13
I think the Voicing parameter is useful.	5.00	5.00	5.00	13

What adjustments would you make to the UI, if any?

progress bar or other ui element while the chords are being generated. Something other than the midi file path as an icon when you load the melody.

Replacing complexity with labelled levels instead of numbers (i.e. describing what the complexity level did)

I think it just needs more. One very slight issue I was having was that it would always generate a G7 chord as the last chord, and it didn't sound nice. Maybe adding a function to filter out unwanted chords might be one possible thing to add.

None. The UI was extremely user friendly and I had no issues with the features at all.

It would be interesting to substitute chords given by the generator while keeping majority of the chord progression if one of he chord in the chord progression would want to be replaced by the user

UI is very easy to use/understand. Could use more visuals such as displaying the MIDI grid with the underlying chords

Could you see yourself downloading this VST if it was free, open source software? Why?

yes, super useful. Fun to try different chords

Definitely. I go through phases where I don't touch music for 1-2 years, then I do it for 1-2 years, then that cycles. Whenever I get back into it, I have to reteach myself some things about theory. Using this software as a educational tool for myself would be useful

If I were to see myself producing music in the future, I would definitely consider downloading VST. It's a nice tool that can help just figuring out what chord progressions sound good. I think this software has a lot of potential!

Yes! This software is genuinely so useful for composing. There are so many music enjoyers who do not know a ton about chord progression. This environment is an AMAZING way to teach this, as well as helps with arranging being able to just put the key and melody in, and a full set of chords can be generated, as well as easily edited and given multiple options for creativity as well. This could even be used by experienced composers who already know this stuff, making the process a little bit easier, and they can make changes where they want.

its fun to play around with and could b useful if you were needing a et of chords for a melody but had 0 ideas and needed some inspiration

Could you see yourself downloading this VST if it were commercially available for $5 or less? Why?

Yes, that's probably about the right price

Yes, for the reason above

I would download this either way. Free or not, as I explained before, the software would still be useful!

I think I could. For $5, the value gained from this is amazing. Most people would get this knowledge at a much steeper price, so to have access to being able to generate chords and learn from that for just $5 is an amazing deal.

Yeah, although I usually purchase apps hat are n the lower scale cost wis and it would be a good investment

Yes, $5 isn't too expensive, however I think most people would want more options such as additional styles, chord voicing options, etc. if they're going to pay

If you owned this VST, how often would you use it?

once a month? not sure

Probably only a few times a year

I don't think I would use it every time I opened up a DAW like Ableton, but I would definitely keep it on hand while producing whatever music is in my brain.

I do not think I would use it on a day-to-day basis, but that is because I do not do music-related activities, nor would I have a use for it day-to-day. When I would use it for arranging/composing though, I feel I would use this very often, as would those who regularly arrange/compose.

Whenever i needed inspiration while making music I would definitely use his chord generator

If you owned this VST, what would you use it for?

songwriting/building an arrangment

Mostly for inspiration, or help when I'm stuck and need ideas

I think I would use it for generating a chord progression, and then if I like it, I'll stick with it. I'd most likely modify it a little to a chord progression that my musical ear would prefer, but overall, it's a great tool to help get me started on whatever music I'm producing.

Arranging and composing. I think there is potential for it to be used as an educational tool, but the education gained from it would need some improvement. Explanations as to why these chords were chosen would aid a lot in this, as for right now the only way to learn is to just recognize patterns in chords.

I would use it to help come up with different chord progression for melodies that I would come up with on my own

What improvements or additional features would you like to add to the VST?

A button to preview the chords (audio) without having to add to the sequence

Adding modes beyond just major/minor (adding dorian, phrygian, etc). Adding ability to place chords whereever in the melody, instead of being rigidly locked into how many chords per bar there will be for the whole thing.

I think a slight improvement would be a shorter midi generating time. As I was going through the tasks, I started to realize how long it was taking for the file to generate. Some cool features could be filtering out certain chords (as mentioned before). Maybe even adding more variety for the type of chords like diminished or augmented chords. I feel it would be too complicated as more details are added, but I definitely think this software has a lot more to offer.

Most of the improvements would be expanding on the project more, which I think is very well doable and could make this unique app even more special. Some suggestions (I am sure time was a big constraint so these were not expected, just some cool ideas): 1) Try adding in more genres to choose chords from. 2) Expand the number of chords to be done. 3) Expand parameters (can include things such as danceability, etc. (there are many song datasets that have other ratings they are given you can find on places like kaggle)..

Only just the editing of chords in a given progression would be kind of neat

Any final comments?

very cool!

I really love this idea! I'm interested to see if it ever grows more!

Excellent job!!!!!

All Good :)

Add funk as a style!

Thank you!