Algorithmic Composition of Two-Part First Species Counterpoint

Ryan Yonek

April 19, 2024

Contents

1 Abstract

The field of Algorithmic Composition has been gaining traction in recent years as both a compositional and educational tool. From generating new music to modeling old styles, computers have been shown to have great potential for representing and demonstrating musical concepts. Since Species counterpoint was developed in the 15th and 16th centuries and has established rules and conventions, this project aims to model this style using Algorithmic Composition rather than try to compose something new. The goal of this project is to experiment with generating First Species counterpoint by using Magenta and recurrent neural networks (RNNs) to determine if Machine Learning can be applied to generate music with as strict of rules as Species counterpoint.

2 Introduction

Species counterpoint is a 16th-century style of music in which a melody called a cantus firmus is the basis for other melodic lines to be written. Any new lines based on the cantus firmus are called counterpoint. There are a total of five Species. Starting with First Species, they get increasingly more complex. Writing in Species counterpoint requires a strict adherence to a set of both melodic and harmonic rules.

This research contributes to the field of of music generation using Machine Learning and can be transformed into an educational tool for students needing additional counterpoint examples. As seen in the "Creating Training and Test MIDI Sets," there is a small number of training data available. Therefore, having the ability to generate more correct examples is invaluable to additional research in machine learning and counterpoint and for educational purposes for music students.

The idea of this project came about from Kelber (2021) [9]. This project utilizes Magenta Recurrent Neural Network models [13] to train and test sets of two-part MIDI data in the style of First Species counterpoint. Before discussing the project design and results, it is necessary to learn the terminology and rules of the style of music that will be modeled.

3 First Species Counterpoint

Counterpoint is not the same as harmony. Counterpoint has harmonic elements, but the distinction between the two is the importance of melody in counterpoint. Harmony is concerned with the vertical element of the music while counterpoint is concerned mainly with the horizontal and only partially with the vertical. In counterpoint, each melody is of equal importance. These melodies come to together to create polyphony, or harmony created by independent melodies.

Species counterpoint was introduced during the 15th and 16th centuries, with the rise of Giovanni Pierluigi da Palestrina (1525-1594). Palestrina wrote exclusively a cappella (only voices) and polyphonic music. The style of Palestrina can be broken down into a few key parts. The first part of Palestrina's style is the use of the C Clef to represent the notes on a staff.

3.1 C Clefs

A **clef** is a symbol used at the beginning of a 5-line music staff to determine what the position of a note on the staff represents. Clefs are used to maintain the majority of singer's vocal range within the 5 lines of the staff, so it is easiest to read. The C Clef is a type of clef that locates middle C (C4), hence the name. The C Clefs are divided into the four vocal parts for SATB: soprano, alto, tenor, bass (soprano being the highest). By shifting middle C higher on the staff, more of the staff can accommodate lower pitches, making it easier to read for tenor and bass singers and vice versa for alto and soprano singers. Figure 1 shows the different types of C Clefs used in the Renaissance Period.

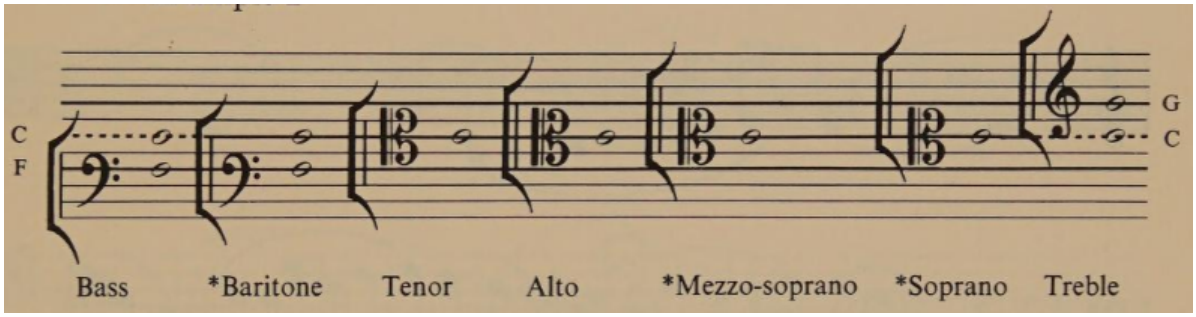


Figure 1: Clefs on the 11 lines of the bass clef, treble clef, and middle C in between [7]

*The baritone, mezzo-soprano, and soprano C Clefs are no longer used today and have been replaced by the bass and treble clefs. For the purpose of this project, only the bass, alto, and treble clefs will be used.

In Species counterpoint, two or more voices sing together. In order to distinguish the voices, each voice gets its own staff and designated clef. The staves are lined up with one another to create polyphony. The melodies that create this polyphony are much different in the Renaissance Period than what we see today. The melodies sung by voices in the 15th and 16th century were organized into modes.

3.2 Church Modes

A **mode** in music is a transposition of the same notes of a scale. For example, starting a C major scale on D creates the Dorian mode. The music of Palestrina was sacred and followed in the footsteps of the Pope Gregory the Great's system of modes which stretched from the early Medieval to the 12th century. This system contained 16 authentic and plagal modes. After more than 300 years, the authentic and plagal distinction became unimportant, and the Gregorian church modes evolved into just 6 polyphonic church modes. The fifth scale degree of each mode is the **dominant** and is considered the note of activity. The **final** of the mode is the first and last note of the mode. The polyphonic modes only appear in counterpoint in white-key form (cantus naturalis) and with one flat, in transposed form (cantus mollis) [7]. Figure 2 shows the 6 modes and their transpositions based on what we now consider the notes of C-Major and F-Major respectively:

The symbols under a few of the notes in the Dorian mode refer to an idea called **Musica Ficta**. In short, Musica Ficta is the use of accidentals by the performer to raise or lower the written pitch. These practices will be further explained in section 2.5.1 Melodic Rules.

After deciding on the clefs of the voices and the mode the counterpoint follows, it is time to use a cantus firmus. The first melody to every Species counterpoint is a cantus firmus and is the basis for which all counterpoint from any of the polyphonic modes is derived.

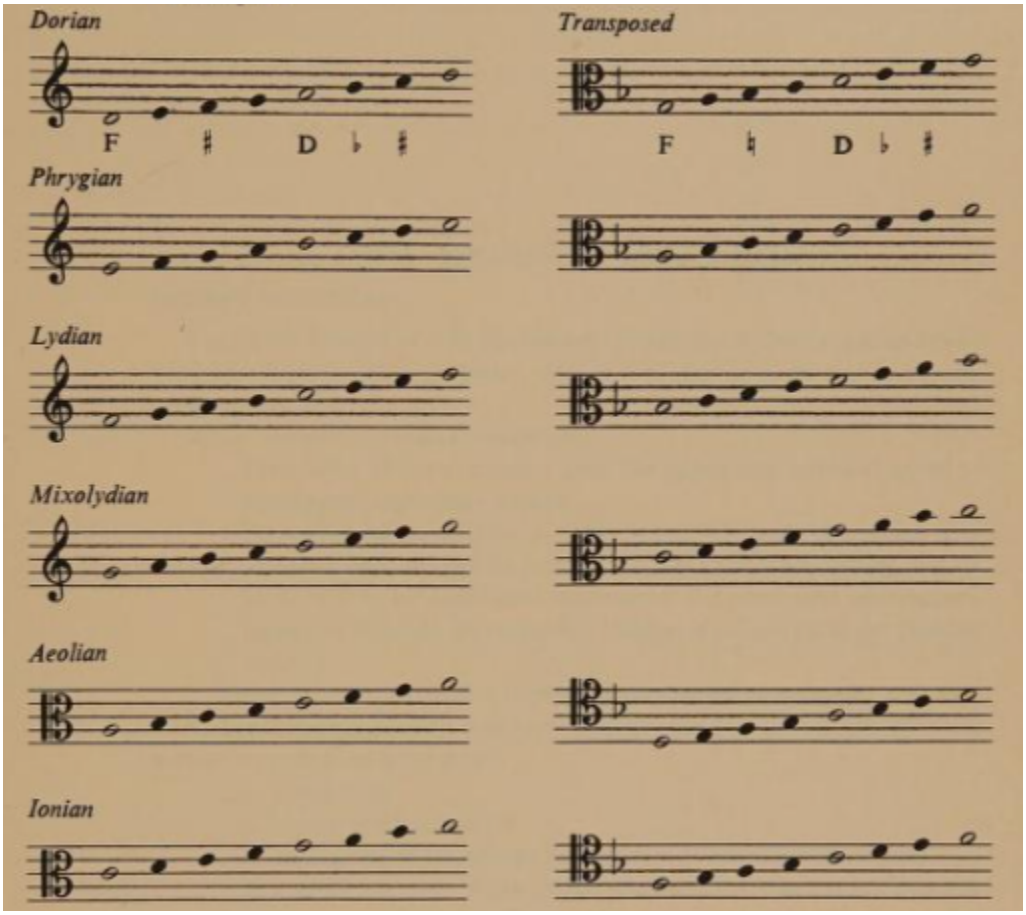


Figure 2: The 6 polyphonic modes and their transpositions [7]

3.3 Cantus Firmus

The word "discantus" was used prior to "counterpoint". "Cantus contra cantum" translates from Latin as "melody against melody". The **cantus firmus** (C.F.) is the first melody of any counterpoint. Figure 3 shows a few examples of cantus firmi:

There are a few distinct parts of the cantus firmus. The C.F. must start and end on the final of the chosen mode (e.g. "D" in Dorian or "E" in Phrygian). It is a melody, so the most important quality of any cantus firmus is that it is easy to sing. This means that it should be primarily step-wise and only contain a select few leaps which will be discussed under 2.5.1 Melodic Rules. A cantus firmus must also have a cadence to signal the end of the melody. In every mode, this cadence must be the second note of the mode down to the final. Every cantus firmus ends the same way.

Now that the clefs, mode, and cantus firmus have been established, it is time to write

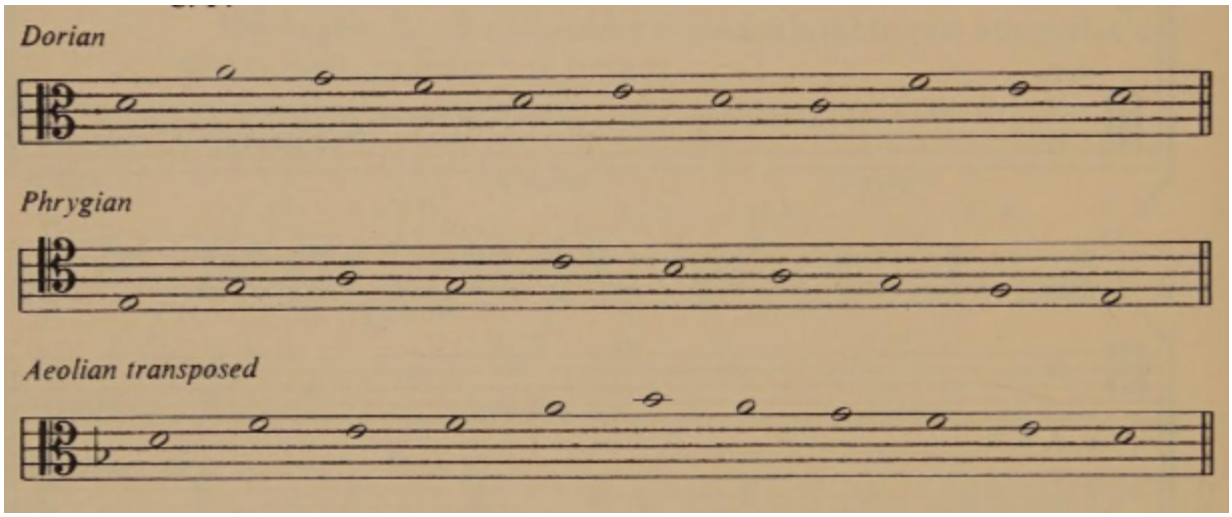


Figure 3: Three examples of cantus firmi in the Dorian, Phrygian, and Aeolian transposed modes [7]

counterpoint. In First Species counterpoint there are several melodic and harmonic rules which are generally adhered to when writing counterpoint [7].

3.4 Rules

First Species counterpoint in two parts is note-against-note counterpoint. In modern times, this will look like a whole note against a whole note per measure. This one-to-one relationship highlights the equal importance of each part which also means that each melody must follow the melodic principles of the style [8].

3.4.1 Melodic Rules

In Species counterpoint, the melodic line is the most important element of the style. For any First Species counterpoint in two parts, these are the melodic rules for the single line of counterpoint paired with its cantus firmus.

1. First Species counterpoint must be predominantly conjunct (stepwise) with occasional, specific moments of disjunct motion (leaps).

2. The conjunct intervals available are major and minor 2nds. The disjunct intervals are minor and major thirds, perfect 4ths and 5ths, minor 6ths ascending, and perfect

octaves. The intervals of an augmented 2nd, major 6th, major 7th, minor 7th, and tritone (A4 or d5) are forbidden in this style.

3. Having more than 2 leaps in the same direction is considered poor.

4. The following intervals with one note in between should not be in continous up or down motion: major 7th, any interval more than an octave, an augmented 4th (a diminished 5th descending is fine), a minor 7th composed of two perfect 4ths (a perfect 5th and a minor third descending is good).

5. When leaping by at least a perfect 4th, the notes preceeding and succeeding the leap must move in contrary motion to the leap.

6. One repeated note is acceptable, but should be avoided in general since it takes away from the independence of both lines.

7. One should avoid monotony by not repeating the same patterns of notes and utilizing an interesting contour with preferably both a singular high point and low point of the melody.

8. The vocal ranges of each voice part can be seen in Figure 4:

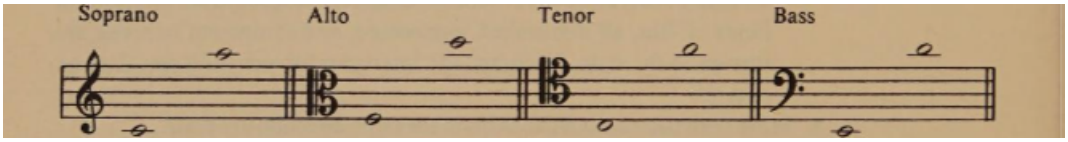


Figure 4: Vocal ranges of each part in SATB [7]

9. Musica Ficta should be observed in several circumstances. The first is using a 'B-flat' or sometimes an 'F-sharp' to avoid an augmented 4th or diminished 5th by a direct leap. This should be applied to the second note. Secondly, a 'B' between two 'A's should be flattened. Thirdly, the 'B' is also typically flattened in a descending scalar sequence. Fourthly, the 'F-sharp' may be used to avoid an augmented second in the Aeoleon cadence. Lastly, in cantus mollis (transposed modes), the 'E-flat' and 'B' serve the same purpose as 'B-flat' and 'F-sharp' respectively [7].

3.4.2 Harmonic Rules

Although harmony is secondary to melody, there are still important conventions to adhere to in order to replicate the style. For any First Species counterpoint in two parts, these are the harmonic rules for the single line of counterpoint compared to its cantus firmus.

1. First Species counterpoint in two parts contains one whole note paired with each whole note of the cantus firmus.

2. The intervals between these two voices must always be consonant. The two types of consonances are imperfect and perfect. Perfect consonances include the unison, perfect 5th, and perfect octave. Imperfect consonances include the minor 3rd, major 3rd, minor 6th, and major 6th. All other intervals, including the perfect fourth, are considered dissonances in this style.

3. The opening interval of the counterpoint must be a perfect consonance. The counterpoint is above the cantus firmus, this interval can be any perfect consonance, but if the counterpoint is below, the perfect 5th music be avoided in order to maintain the mode of the cantus firmus since the final of the cantus firmus would become the dominant of the counterpoint mode.

4. The unison can only be used in the first or last measure.

5. In general, imperfect consonances are preferred over perfect ones.

6. With regards to spacing, voices should not be more than a 12th apart.

7. At most 3 consecutive 3rds or 6ths in similar motion (parallel or direct) are allowed.

8. In this style of contrapuntal writing, voice crossing is common and occasional voice overlap is allowed.

9. Parallel 5ths and octaves are not allowed due to the lack of independence of parts. Consecutive 5th are only allowed in contrary motion or from crossed voices.

10. Direct motion to any perfect interval should be avoided.

11. The last two notes of the cantus firmus must create a cadence. The penultimate note of the counterpoint must be the leading tone or seventh note of the mode resolving into the final of the mode. This means that the penultimate note must be sharped in Dorian, Mixolydian, and Aeolian. The seventh note in Phygrian may remain since the upper resolution in the cantus firmus acts as the leading tone [7].

Now that the rules and terminology of First Species counterpoint are defined, the next section will introduce Algorithmic Composition, the general field for generating music through computer software.

4 Algorithmic Composition

Algorithmic Composition has become an enticing field for computer scientists over the last century. Using artificial intelligence to generate music is a tool that many composers have utilized in recent years. Algorithmic composition can also be used to model or analyze music in creative ways as well.

4.1 Related Research

In the more specific realm of algorithmic composition of counterpoint, many different algorithms have been used for various purposes. One of these algorithms is called a **Markov chain**, which consists of a sequence of events called states in which the probability of a state depends on the previous state. Due to how sequential Markov chains are, they are especially useful for generating melodies. One of the first systems in Algorithmic Composition ever described was created by Pinkerton (1956) [4] called the "Banal Tune Maker." This system analyzed the transitions of 39 nursery tunes by hand to create a transition matrix. Unfortunately, Markov chains risk plagiarizing often because they reuse previous material. However, some results are novel and can be considered creative in the product category.

Another system used in the field of Algorithmic Composition is a **Generative Grammar** which "is composed of two alphabets: terminal symbols and non-terminal symbols (or variables). A set of rewriting rules is given over the union of these two alphabets, that allow

to transform variables into other symbols (both variables and terminals) [4]." Any sort of musical information, but especially chord sequences, can be transformed using Generative Grammars. This system can be used for both music generation and analysis. For example, "Steedman (1984) compiled a Generative Grammar to describe Jazz chord sequences [4]."

Music theory rules fit especially well in **Rule-Based Systems**. Especially when using Constraint Programming, computers are able to learn rules quite well. These systems typically require an input before creating something. Section 4 will explore an example of this, Schottstaedt's [3] approach to generating First Species counterpoint.

Another technique has become quite popular for generating images and video, processing language input, and generating music. This technique is **Neural Networks/Deep Learning**. The approach in this project is based on this technique and Section 5 will dive deeper into this approach.

Regardless of the system or algorithm, when generating art through computational means, it is important to consider whether the art is maintaining its creative nature. Many musical systems such as Species counterpoint were created through human creativity. The question remains whether computers can match this human creativity.

4.2 Compositional Creativity

Computational Creativity is defined as "The philosophy, science and engineering of computational systems which, by taking on particular responsibilities, exhibit behaviors that unbiased observers would deem to be creative [4]." The two main elements of creativity are novelty and value. The categories of creativity to be evaluated on being original or useful can be better defined into the four P's: person, product, process, press. Person refers to the agent and its personality, flexibility, and intelligence. Product refers to the artifact or creation of the agent. Process is the internal set of actions to complete the creation, and press is the cultural response/view of the creation.

The four P's can be applied to many topics in Algorithmic Composition to assess whether a computer is actually creative. In general, if a human cannot distinguish computer creativity from human creativity, the computer has achieved a satisfying level of creativity. The goal of this project is create something that is indistinguishable from the human equivalent. If

a computer can generate correct First Species counterpoint based on conventions created by humans, than the computer is demonstrating a baseline level of compositional creativity. However, before evaluating this project's approach specifically, it is important to address other approaches in the same field in order to compare and learn about the complexities of the field.

5 Intuitive Approach

With the heyday of Species counterpoint having been half a millennium ago, the goal is not to generate something novel. Rather, the quality/value side of compositional creativity is the most important function of this implementation. There are many sides to algorithmic composition but the two approaches that make the most sense for generating First Species counterpoint are a rule-based system using constraint programming and a machine learning system using a neural network. The next section will explore the intuitive, rule-based approach for generating First Species counterpoint.

5.1 Rule-Based/Constraint Programming System

Species counterpoint is built on rules so it only makes sense to implement Algorithmic Composition of this style using a rule-based system. Rule-based programming is better for composing music due to the flexible nature of rule-based programs in which new rules can be added and removed unlike procedural programs. By implementing these rules in this modular system, the melodic, harmonic, and aesthetic parts of this music can be formalized by a computer [3].

5.1.1 Organization of Rules

These rules can be better defined using Constraint Programming (CP). The elements of CP are variables (unknowns) and constraints (relations) between the variables. Variables in the case of Constraint Programming are more similar to the mathematical unknown in an equation rather than a variable in computer science that can be given a value and altered at any point in the program. A Constraint Satisfaction Problem (CSP) contains these variable

and constraints and a constraint solver would find a solution to the variable based on the constraint.

In the case of First Species counterpoint, the variables would be the number of whole notes needed to match up with their respective whole notes from the cantus firmus. Each constraint would be one of the rules. These constraints can be both melodic and harmonic.

5.1.2 Weighted Penalties

The criteria for adding a note are determined by penalties for each choice which are measured loosely by Fux [8]. In First Species, some choices are strictly forbidden while others are merely discouraged. This ambiguity of punishment for mistakes was explored by William Schottstaedt. Figure 5 shows a list of some of the constraints and their corresponding penalties Schottstaedt used in his approach to Constraint Program and Species Counterpoint [12].

Very bad infractions (penalty is 200):

1. Avoid direct motion to a perfect fifth
2. Avoid direct motion to an octave.
3. Voices should not move outside certain ranges.
4. Avoid a unison on the down beat.

Figure 5: Constraints and their corresponding penalties [12]

Each decision has a penalty. All of these decisions can take a long time depending on the number of notes in the cantus firmus. With a cantus firmus of n notes and 16 possible notes to go with each whole note, doing a brute force approach would take $O(16^n)$ time Big-O Notation. This approach is incredibly inefficient so it is important to find a more efficient approach. [12].

5.1.3 Recursive Searching and Greedy Approach

Schottstaedt [12] uses a greedy approach [5] in his recursive searching algorithm as a more efficient approach to generate species counterpoint. Schottstaedt's greedy choice was to al-

ways take the note with the smallest penalty. Using recursion, this approach makes branches of "best-first" solutions. Unfortunately, this approach does not have the greedy choice property because an optimal choice early on does not mean optimal choices later for every cantus firmus. With the variability of cantus firmi, the greedy approach will not be an optimal approach, but it will be more efficient than a brute-force approach.

If a problem occurs later where the penalty gets to be too large where it surpasses the best solution, it is possible to return to original note and make the second-best choice and continue. This original note will probably be the second or third note of the cantus firmus because the first note can only be a perfect consonance which leaves only 2-3 choices. The second or third note of a counterpoint is less constricted. Once all of the best solutions to either the second or third have been completed into full counterpoint solutions, the program is complete [12]. Figure 6 shows a completed counterpoint through the process of constraint programming and with the greedy choice related to the penalty of each choice.

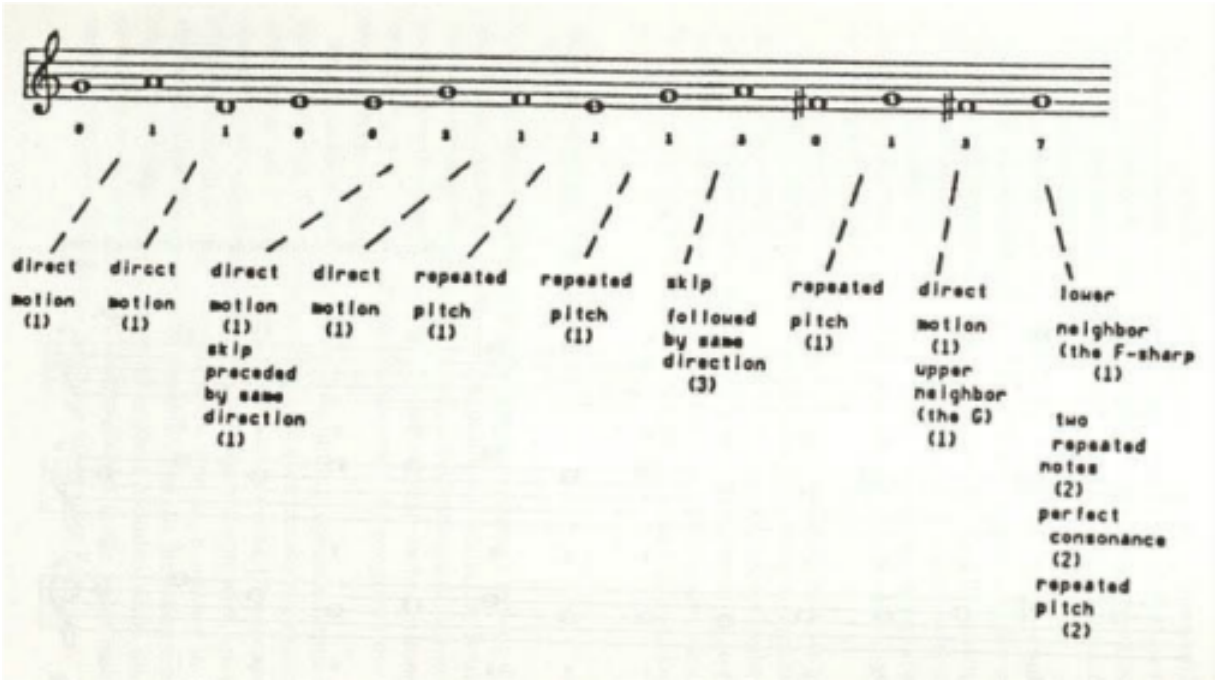


Figure 6: Line of First Species Counterpoint with penalties for each choice [12]

6 My Approach: Machine Learning/Recurrent Neural Network

The approach that I use to generate First Species counterpoint is with a model inspired Adiloglu and Alpaslan's model called *NeuroComposer* [1]. This approach is fundamentally different from the rule-based system in the previous section because it actually does not use specific rules at all. Instead, the system models each note as a collection of nodes in an **artificial neural network** (ANN). There are different representations for the input and output sides of the neural network. Each side uses a type of pitch (note) representation, melody representation, and time representation. These representations will be explored in the next section.

6.1 Initial Model of Motivation

In *NeuroComposer* [1], the First Species counterpoint is modeled by a feed-forward artificial neural network. This network is split into two parts, the input side which is where the cantus firmi will be stored and the output side where the counterpoint will be generated initially by the training examples. In Species counterpoint, whether a note fits or not depends on the previous notes. This relationship is defined through time representation, or a sequence of notes.

Music happens in time and in the case of First Species counterpoint, in a series of whole notes. This time relationship between notes needs to be represented in the ANN so the network can correctly assess each note at a time and compare it to other notes in the sequence. There are several types of time representations. The first is called the **time window representation** which divides a melody into smaller time frames called "windows". The **sliding window representation** allows the window to move one note at a time as more notes are added. Like the penalties from Section 4.1.2, weights are obtained and altered through training a set of examples. These weights determine the importance of each note within the window. In the **decaying time representation**, all notes before the input get a value between 0 to 1. The most recent notes have the highest coefficient and the

notes farther away "decay" and get lower coefficient values. Since the earlier notes have a greater coefficient, this value represents their influence on the input note. Thus, more recent notes have more influence on the upcoming output note. The **decaying time window representation** combines the previous two methods and limits the decay to a window of time which slides as new notes are added to the output counterpoint. The next important element of representing music in an artificial neural network is to be able to represent the pitches themselves.

Pitch representation is how musical pitches are broken down and understood by computers. The first of these is **spectral representation** which uses the spectrum of the acoustic signal to represent the note. This is type of representation is not effective for comparing notes using music theory rules such as counterpoint so the spectral representation for pitches will not be used.

The second type of pitch representation is **interval representation** in which set of pitches is represented by successive intervals between the notes. Each sequence of pitches has a reference note in which all of the future notes are based on. This reference note can be the first note of the melody, the tonic or the dominant note of the mode of the piece. Interval representation satisfies the rule of invariance under transposition, since the notes of a sequence can be transposed and maintain their interval properties in the mode.

The third type of pitch representation is **pitch height representation** which is created from an analysis of the fundamental frequency of the pitch. Each pitch has a value related to its MIDI standard from [-1,127]. Moreover, two notes that share the same pitch value (letter name and accidental) have different pitch height values by multiplication. This allows the system to distinguish between higher and lower notes which is especially important when there is voice crossing and intervals become inverted. Unfortunately this representation does not satisfy the rule of invariance and thus the sequence of intervals cannot be maintained through transposition.

The last pitch representation is **pitch class representation** in which the octave multiplications of the pitch from the pitch height representation are removed. This means that only the value from [-1,127] is used for the pitch representation of the note.

In this study specifically [1], different representations were applied to the input and output

parts. The **pitch height representation** technique was chosen for the input, because it has the advantage that it allows easy control of the range of the pieces. As seen in the melodic principles of First Species counterpoint, vocal range must be maintained. In this representation, each input node has 13 Nodes as seen in Figure 7.

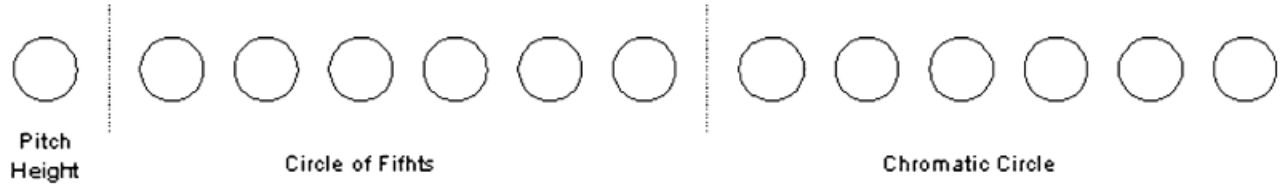
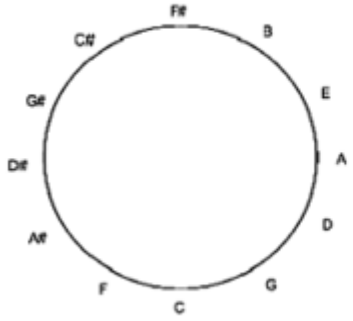


Figure 7: 13 Nodes for the Input Pitch Representation [1]

The **chromatic circle** and the **circle of fifths** representations in Figure 8 are used in addition to the categorical pitch height representation since it does not satisfy the rule of invariance between transpositions.



Chromatic circle

CC Tone	Representation					
C	1	1	1	−1	−1	−1
C#	1	1	1	1	−1	−1
D	1	1	1	1	1	−1
D#	1	1	1	1	1	1
E	−1	1	1	1	1	1
F	−1	−1	1	1	1	1
F#	−1	−1	−1	1	1	1
G	−1	−1	−1	−1	1	1
G#	−1	−1	−1	−1	−1	1
A	−1	−1	−1	−1	−1	−1
A#	1	−1	−1	−1	−1	−1
B	1	1	−1	−1	−1	−1

Figure 8: Circle of Fifths (left) and Chromatic Circle Binary Representation (right) [1]

The output notes of the counterpoint are represented by a **pitch class representation**. Since the pitch class representation does not include the octave, a separate set of nodes are used to indicate the octave value. Therefore, each output note is composed of 22 nodes: 6 from the circle of fifths, 6 from the chromatic circles, and 10 octave value nodes as seen in Figure 9.

Now that we have covered the elements of an example artifical neural network implementation of First Species counterpoint. This project dives deeper into the concepts of Machine

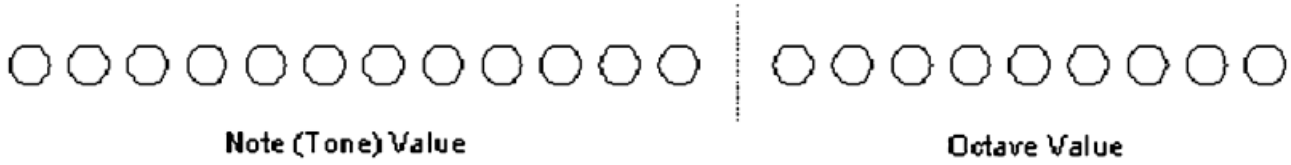


Figure 9: 22 Nodes for the Output Pitch Representation [1]

Learning and Neural Networks so it is important to establish a strong understanding before going through the implementation.

6.2 Machine Learning Model

Machine Learning (ML) is a key sub-field in Algorithmic Composition. In the context of ML, learning is defined as "any computer program that improves its performance at some task through training [11]." The goal of this project is to have a computer program learn over time how to write First Species counterpoint. A computer program learns through experience gained from some input of tasks. This experience helps it perform its own tasks which have some level of accuracy. This accuracy increases as the level of error between the computers output and the evaluation/testing output decreases.

6.2.1 The Learning System

The process of Machine Learning can be implemented by creating a **learning system**. This system consists of several parts. The first part is the **learning experience** which consists of three parts. The three parts of the learning experience are the knowledge to be learned, a representation of this knowledge, and a learning mechanism. A learning mechanism has to provide feedback to the network in each step of the training experience in order to credit which parts of the network have control over the final results.

The next step is to choose the **target function**. This function represents the knowledge that is learned and how that knowledge will used by the program. This function is based on the set of tasks. The goal of the program is to get as close to the target function as possible through adjusting the weights of its own function.

The most popular way to represent the learning program to describe the target function

is through **artifical neural networks** (ANNs). The goal is for this network to represent an approximation as close as possible to the target function. The more complex the network, the more training data needed in order to fit the network.

In order to learn the approximation for the target function, a set of training examples is required. Each example describes a state and value. Each step in each training example has a weight applied to it so an important element of creating an approximation is adjusting the weights for each step. **Perceptrons** are involved in this process of adjusting weights. These adjustments are made iteratively until the squared error between training values and the approximate values from the network is minimized. These adjustments can be made through training rules such as the Perceptron Training Rule or Gradient Descent Training Rule.

Machine learning models cycle through 4 generic models during training. These 4 models are Performance System, Critic, Generalizer, and Experiment Generator. The **Performance System** takes an instance of a new input and using the learned evaluation function, and outputs a history of the solution to the problem. The **Critic** takes that history and produces training examples of the target function. The **Generalizer** takes the training examples and returns estimates of the target function based on each example. The **Experiment Generator** then takes this new estimate and creates a new problem for the Performance System to solve. Figure 10 shows this system [11].

The artificial neural network is one specific representation of Machine Learning that is commonly used in music generation. The next few sections will explore the composition, parts, and algorithms within ANNs.

6.3 Artificial Neural Network (ANN)

An **artificial neural network** (ANN) is a representation used for hypothesis spaces. Learning algorithms utilize the structure of this network to organize the search through these spaces. ANNs consist of input, hidden, and output units. These units together typically form a directed acyclic graph. Each vertex on the graph corresponds with the output of a single network unit. Neural networks have a significant biological motivation since ANNs aim to model many parts of human learning systems by creating a complex graph with many

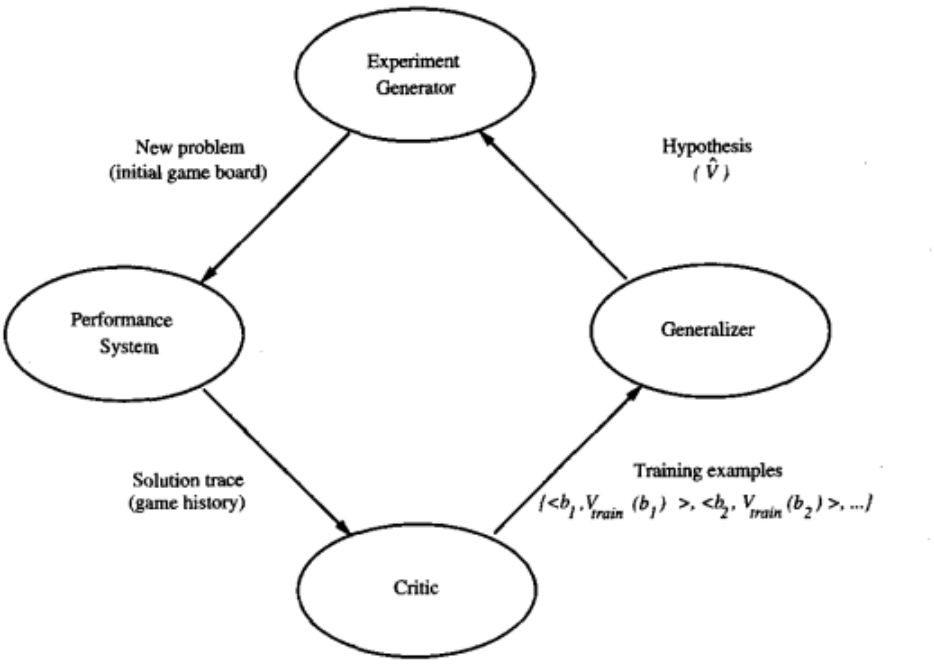


Figure 10: Machine Learning Generic Model [11]

interconnected units.

6.3.1 Biological Motivation

ANNs were originally inspired by biological systems built on webs of interconnected neurons, hence the name "neural" network. "The human brain, for example, is estimated to contain a densely interconnected network of approximately 10^11 neurons, each connected, on average, to $l0^4$ others [11]." ANNs attempt to capture the parallel computation of the information processing of biological neural systems. ANNs are generally constructed through densely-connected simple units where each unit takes one or more inputs and produces a single output. This output may become the input for one or multiple other units in the network.

6.3.2 Perceptions

One type of unit in ANNs is the **perceptron**. A perceptron takes a vector input, calculates a linear combination of the inputs, and outputs either a 1 or -1 depending on some threshold. Due to the binary nature of the perceptron output, a single perceptron can be used to

represent true (1) or false (-1). The weights within the linear combination can determine the type of boolean function represented.

6.3.3 Training Rules

ANN training typically takes a long time (hours) due to the complexity of the interconnected network. One of the main factors that determines the training time is the number of weights in the network. The more weights there are, the more iterations it will take to modify them to create the best possible approximation of the target function. The values of these weights are fine-tuned through training rules.

One of these training rules is the **perceptron training rule**. After starting with random weights, the perceptron is iteratively applied to each training example and the perceptron weights in its linear combination are modified. Once all of the training examples are classified correctly by the perceptron, all of the weights are modified. The only limitation to the perceptron training rule is that the training examples must be linearly separable and the perceptrons must be thresholded.

The other training rule for adjusting ANN weights is the **delta training rule**, also known as the least mean squares (LMS) training rule. The delta training rule converges toward the best approximation of the target function through **gradient descent**. Gradient descent searches the hypothesis space until it finds the best weights for the training examples. Gradient descent is the basis for the **back propagation algorithm**. Unlike the perceptron training rule, the delta training rule trains unthresholded perceptrons. The delta training rule measures the error between the approximate weight vector of the network to the training examples. The hypothesis space is constructed on the axes of weights. Gradient descent determines the steepest weight vector on the hypothesis space. The weights are altered until the global minimum on the hypothesis space is found. The networks this project uses employ back propagation, and therefore, the delta training rule as well.

6.3.4 The Back Propagation Algorithm

Back propagation is used on nonlinear, multi-layer networks. Perceptrons only work with linear decision surfaces while the gradient descent algorithm works with nonlinear decision

surfaces. The back propagation algorithm learns the weights for a network of multiple layers. This algorithm uses gradient descent to minimize the squared error between the output values of the networks and the target values for each output. Since there are multiple outputs, there is a chance for multiple local minima. Back propagation still proves to be quite successful even though it is not guaranteed to find a global minimum on the error surface [11].

6.4 Recurrent Neural Network (RNNs)

"NeuroComposer" [1] implements an artificial neural network through MIDI message sequences and the analysis tools called Aspirin & Migraines C Libraries. Although this was fairly effective, this project aims to use more modern software and experiment with recurrent neural networks (RNN). RNNs are a type of artificial neural network but they have some additional unique characteristics that apply well to music. Recurrent neural networks are also directed graphs and can be trained with back propagation. But RNNs also apply time series data to outputs. The network gradually unfolds through back propagation and eventually the final weights can be determined by taking the mean of the copies of the weights. This project uses some of Magenta's RNN models to attempt to generate First Species counterpoint.

6.5 Software

After exploring many options for experimenting with neural networks, Magenta was the most interesting software to experiment with. Most software I explored was not made for MIDI inputs and it would have required too much development to accomplish. Magenta has existing open source models available for software development and individual model training and testing.

6.5.1 Magenta

"Magenta is a research project exploring the role of machine learning in the process of creating art and music [13]." Magenta uses Python for the back-end software and TensorFlow.js to model graph data from the recurrent neural networks in the browser. The two

Magenta models used during this experimentation were Melody_RNN and Polyphony_RNN. Melody_RNN is a Magenta Recurrent Neural Network model used to generate a melody. It takes single-line MIDI files as inputs for training and testing. Polyphony_RNN is a Magenta RNN model used to generate polyphony. It takes polyphonic MIDI track inputs for training and testing, along with priming melodies and pitches for additional testing.

6.5.2 TensorFlow

TensorFlow is an auxiliary technology used by Magenta models to plot data and graphs for their RNNs. During the training and evaluation of the RNNs, using a command line script, the TensorFlow graph and data can be accessed in a temporary hosted server in the browser. See Figure 11:

```
tensorboard --logdir=/tmp/polyphony_rnn/logdir
http://localhost:6006 to view the dashboard
```

Figure 11: Command line script to access TensorFlow data [13]

Using TensorFlow, the training and evaluation data is plotted. One of TensorFlow's graphs shows the loss-per-step. Back propagation and gradient descent aim to minimize this loss per step in order to improve accuracy. Figure 12 shows an example of the loss-per-step graph in run 2 of the Polyphony_RNN.

6.6 My Work

TensorFlow also shows graphs for both the training and evaluating steps going through the RNN. Figure 13 shows a portion of the train graph for run 2 of Polyphony_RNN.

I used the Melody_RNN and Polyphony_RNN models to attempt to generate First Species counterpoint. In order to train the models, a dataset is needed. So I created 52 training files for the network and 18 testing files once the training of the Polyphony_RNN network was finished.

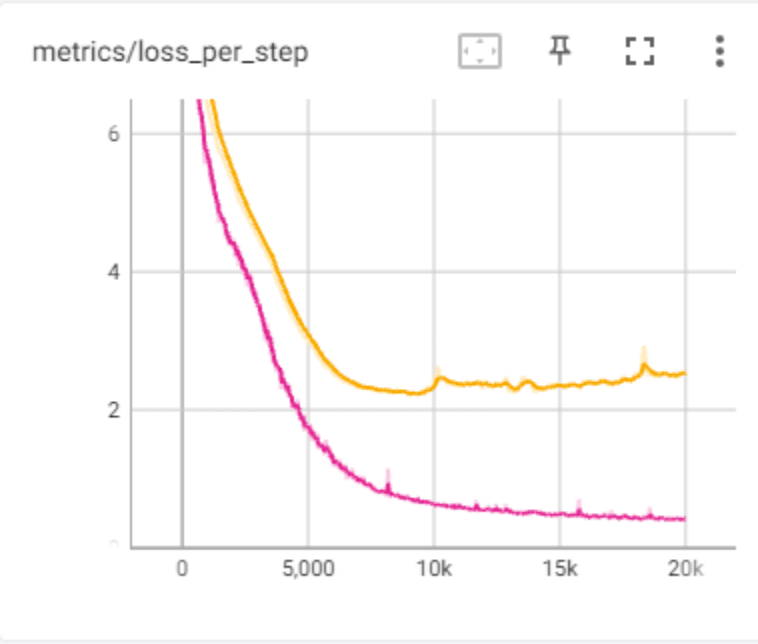


Figure 12: Loss-per-step graph for run 2 of Polyphony_RNN on TensorFlow.js

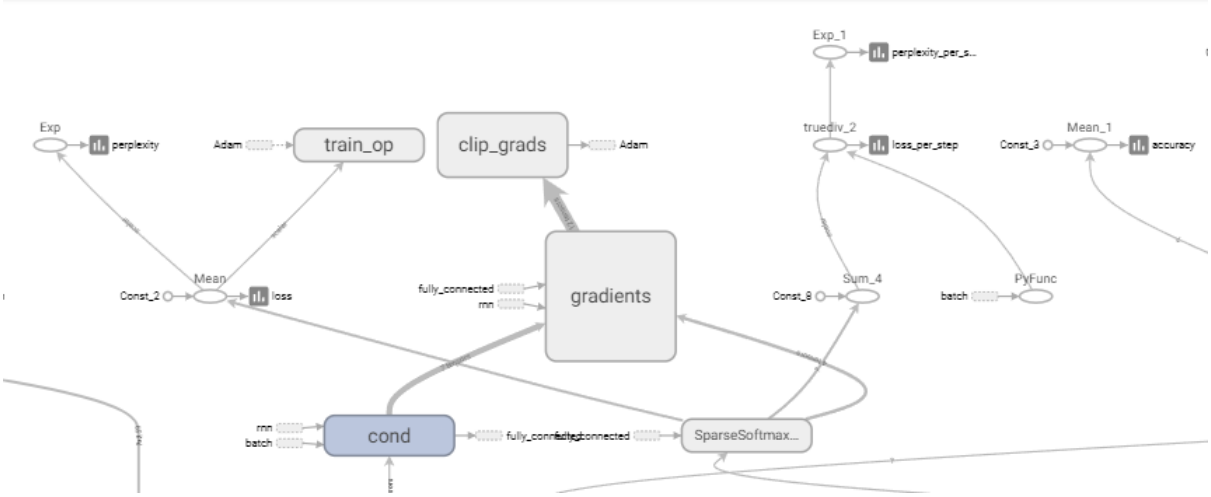


Figure 13: Portion of the Train graph for run 2 of Polyphony_RNN

6.6.1 Creating Training and Test MIDI Sets

To create the training sets for the Melody_RNN and Polyphony_RNN models, I used examples from Fischer and Roberts (1967) [7] and from my Counterpoint class. I notated these examples in the notation software **Finale**. Through Finale, I was able to export each Finale Notation File as a MIDI. I used 26 cantus firmi and composed 52 counterpoints, 26 above and 26 below using my knowledge from my Counterpoint class. Figure 14 shows a file from

the training set.

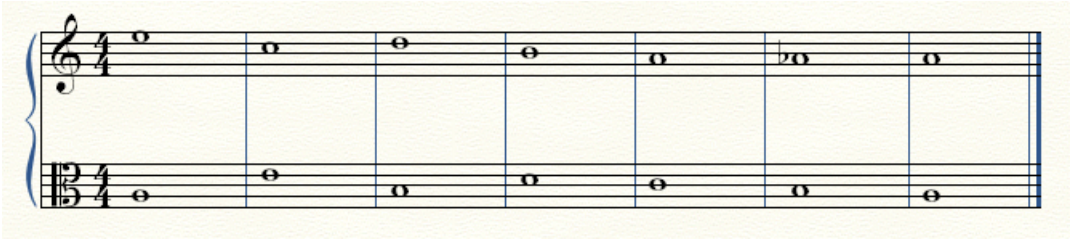


Figure 14: MIDI from Training Set

The test MIDI set I created through Finale as well. I created three test MIDI files for each mode: two with a counterpoint above (octave and fifth) and one with a counterpoint below (octave). See Figure 15 for an example test file.

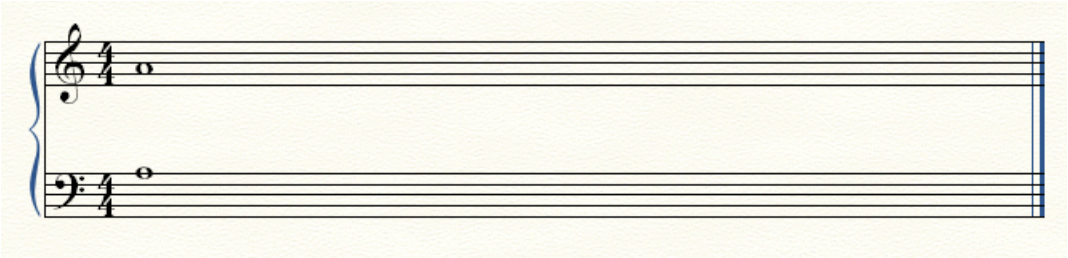


Figure 15: MIDI from Testing Set

6.6.2 Using Magenta Command Line Scripts

I used Ubuntu to run the Magenta command line scripts [13] [6] to train Polyphony_RNN with my training dataset. First I activated Magenta. See Figure 16:

```
source activate magenta
pip install magenta
```

Figure 16: Command Line Script to Activate Magenta [13]

Then I converted my set of MIDI files to **NoteSequences**, which is the format Magenta uses to work with input files. See Figure 17:

Now that the data is in the right format, the dataset can be created. This process converts the 52 NoteSequences into usable files in the training process. See Figure 18:

```
convert_dir_to_note_sequences \
  --input_dir=/home/ryanyonek/FirstSpeciesCtpt/JuniorIS/PolyphonyTrainingSet/ \
  --output_file=/tmp/notesequences.tfrecord \
  --recursive
```

Figure 17: Command Line Script Converting MIDI to NoteSequences [11]

```
polyphony_rnn_create_dataset \
--input=/tmp/notesequences.tfrecord \
--output_dir=/tmp/polyphony_rnn/sequence_examples \
--eval_ratio=0.10
```

Figure 18: Creating a Dataset [11]

Once the dataset is formed, it is time to start training the data in the RNN. See Figure 19:

```
polyphony_rnn_train \
--run_dir=/tmp/polyphony_rnn/logdir/run1 \
--sequence_example_file=/tmp/polyphony_rnn/sequence_examples/training_poly_tracks.tfrecord \
--hparams="batch_size=64,rnn_layer_sizes=[64,64]" \
--num_training_steps=20000
```

Figure 19: Training the RNN with the new dataset [11]

During training, it is important to also be evaluating the data to measure metrics like loss-per-step. See Figure 20 for the altered command line script.

```
polyphony_rnn_train \
--run_dir=/tmp/polyphony_rnn/logdir/run1 \
--sequence_example_file=/tmp/polyphony_rnn/sequence_examples/eval_poly_tracks.tfrecord \
--hparams="batch_size=64,rnn_layer_sizes=[64,64]" \
--num_eval_examples=20000 \
--eval
```

Figure 20: Evaluating the RNN with the 10% of the dataset [11]

Once the long training process is complete, the network is ready to generate polyphony. Figure 21 shows this command line script.

```
polyphony_rnn_generate \
--run_dir=/tmp/polyphony_rnn/logdir/run1 \
--hparams="batch_size=64,rnn_layer_sizes=[64,64]" \
--output_dir=/tmp/polyphony_rnn/generated \
--num_outputs=10 \
--num_steps=128 \
--primer_midi=/home/ryanyonek/FirstSpeciesCtpt/JuniorIS/PolyphonyTestingSet/Above_Aeolian5.mid \
--condition_on_primer=true \
--inject_primer_during_generation=false
```

Figure 21: Generating 10 MIDI files in Run 2 of Polyphony_RNN [11]

6.6.3 Results

After training the Melody_rnn with a set of cantus firmi and counterpoint lines separately, I was hoping that after priming the network with a cantus firmus, it would return something related to the cantus firmus and possibly counterpoint. Instead it generated MIDI files like Figure 22.

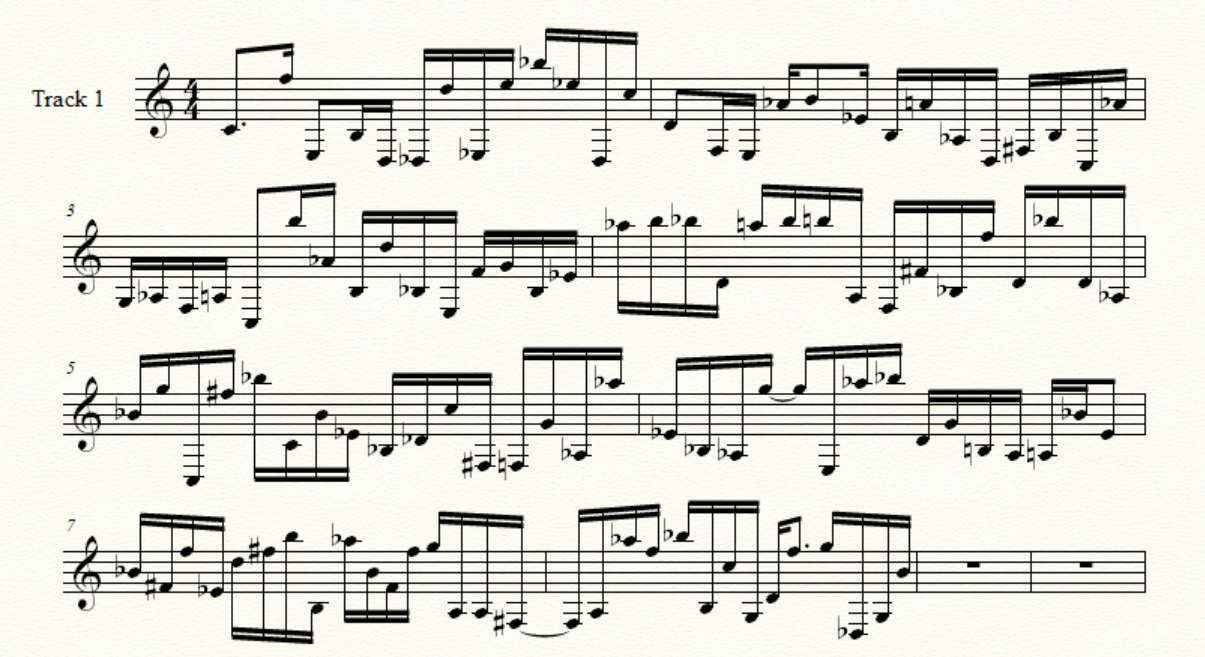


Figure 22: MIDI generated from Melody_RNN

After failing to generate anything close to First Species counterpoint with Melody_RNN, I decided it would be better to use two tracks together in one file and train Polyphony_RNN instead. The first run was trained about 20,000 steps too long and severe overfitting occurred. After doing only 20,000 steps total for run 2, less overfitting occurred. Figure 23 shows one

of the results from the second run of Polyphony_RNN.

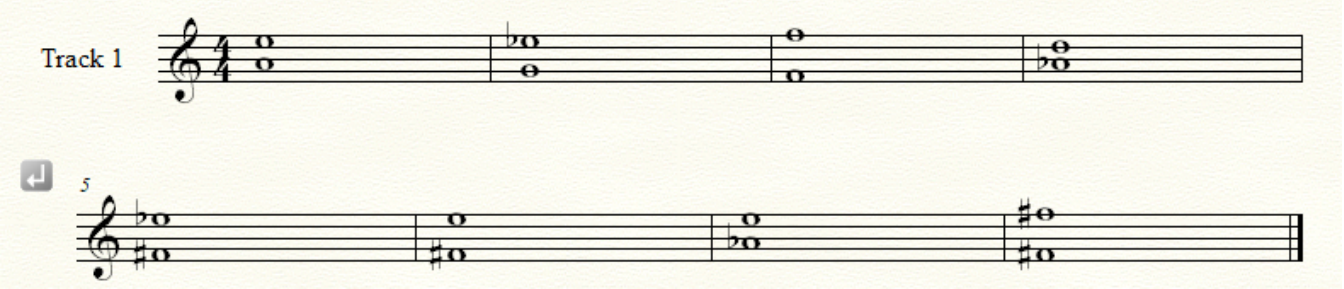


Figure 23: MIDI generated from run 2 of Polyphony_RNN

The generated MIDI files like in Figure 23 contain only whole notes which is a prominent characteristic of First Species counterpoint. Many also contain harmonic intervals of thirds, sixths, and octaves which are common consonant intervals in First Species counterpoint. Although these results are much closer to First Species counterpoint, the results have extra accidentals and contain many dissonances such as seconds, fourths, and sevenths. None of the MIDI files end with a cadence which is an important part of Species counterpoint. The network broke a lot of First Species rules but also incorporated things not explicitly stated such as note length and some consonant intervals. The network did learn some elements of First Species counterpoint from the dataset but not every detail.

7 Conclusions and Future Goals

The most important shortcoming of this experiment is the low amount of training data. Most datasets for training networks as extensive as Magenta RNNs have 1000s of MIDI files. As a result of having only 52 training MIDI files, the data overfit the network and did not teach network enough about First Species counterpoint, even in 10s of thousands of steps.

The function TranspositionPipeline() in the polyphony_rnn_pipeline.py file in the Polyphony_RNN model needs to be adjusted or deleted for the purposes of Species counterpoint training in order to prevent additional sharp and flat notes from being added. If changing this function successfully removes the extra accidentals, the training data has great potential to train the network to be much closer to conventions of First Species counterpoint.

There is a lot of complexity to Magenta's RNN models and there is so much to explore

with each of their models. A future goal of this research is to be able to fully generate all five Species counterpoint styles. Another goal is to be able to train the data to just the right point in order to not overfit.

It seems recurrent neural networks can only be applied partially to represent First Species counterpoint. There was clear learning that occurred through the hours of training, but a lack of training data prevented this model from generating human-level creative examples.

References

[1] Kamil Adiloglu and Ferda N. Alpaslan. A Machine Learning Approach to Two-Voice Counterpoint Composition. *"Knowledge-Based Systems*, 20(3):300–309, April 2007. `https://rave.ohiolink.edu/ejournals/article/329274989` Accessed: April 19, 2024.

[2] Octavio Alberto Agustín-Aquino, Julien Junod, and Guerino Mazzola. *Computational Counterpoint Worlds*. Springer Cham, 1st edition, 2015.

[3] Torsten Anders and Eduardo R. Miranda. Constraint Programming Systems for Modeling Music Theories and Composition. *ACM Comput. Surv.*, Vol. 43(No. 4), Oct 2011. `https://doi.org/10.1145/1978802.1978809/` Accessed: April 19, 2024.

[4] Filippo Carnovalini and Antonio Rodà. Computational Creativity and Music Generation Systems: An Introduction to the State of the Art. *Frontiers in Artificial Intelligence*, 3, 2020. `https://www.frontiersin.org/articles/10.3389/frai.2020.00014` Accessed: April 19, 2024.

[5] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms, Fourth Edition*. The MIT Press, 4th edition, 2022.

[6] Alexandre DuBreuil. *Hands-On Music Generation with Magenta*. Packt Publishing Ltd, 2020.

[7] Irwin Fischer and Stella Roberts. *A Handbook of Modal Counterpoint*. Free Press, 1st edition, 1967.

[8] Johann Joseph Fux and Alfred Mann. *The Study of Counterpoint: From Johann Joseph Fux's Gradus ad Parnassum*. W. W. Norton Company, revised ed. edition, 1965.

[9] Alex F. Zalman Kelber. How I Built an App to Algorithmically Compose Music in the Style of the Italian Renaissance. *Medium*, 2021.

[10] Max V. Mathews and John R. Pierce, editors. *Current Directions in Computer Music Research*. The MIT Press, 1989.

[11] Tom Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[12] William Schottstaedt. *Automatic Counterpoint*, pages 199–214. In Mathews and Pierce [10], 1989.

[13] Google Brain Team. Magenta: Music and Art Generation with Machine Intelligence. `https://github.com/magenta/magenta`. Accessed: April 19, 2024.